Chapter 7. Function or Method

1. Functions

It is the name of a program segment which can be involved from some other point. It may be used as frequently as needed. Basically, a function is a subprogram within a program, created to perform a particular task. It can be called from any part of the program for assistance and execution.

1.1 Two types of functions

- **Built-in or Library functions** – these are the functions stored in the library of any particular programming language compiler may it be Java or any other programming language. To include a specific library function in Java, we have to import the package and the class that defines the function of our need. For example, to the basic input function, readLine( ), we need to import the package and class – java.io.BufferedReader in our program as java.io is the required package and BufferedReader is the class where readLine( ) function is defined.

- **User-defined functions** – these are created by the programmer within a program with a valid name. A function can be made for local use within the class itself where it is declared or it can be made for global use that is it can be used by other classes in other programs.

1.2 Why do we need a function?

The function basically helps in reducing the complexity of the program logic by modularization of the problem. Modularization means breaking down a large problem into a group of smaller sub modules that can work together to solve that problem. Thus, function provides the following advantages in our program:

- Reducing the complex logic into smaller units
- Re-usability of code

1.3 Basic elements associated with a function

There are 3 basic elements associated to the use of a function, they are –

- **Function declaration or function prototype** – it is a declaration of the function that will be used later. If defines a function name with the return type and parameter list. If provides the compiler with the information required for compile-time checking.

```
e.g –     void display (void);
          int largest ( int, int );
```

- **Function definition** – A function is a subroutine that can act on data and then return a value. Each function must be given a name for identification, following the same rules that are applied to identifiers. This is referred to as function declaration.

It has 4 components –

i. return type
ii. function name
iii. list of parameters or arguments
iv. body of the function

- **Function call** – When the execution of the function body is required at some point within a program, it is called or involved. To call a function, the name of the function along with the list of parameters is to be specified ending with a semi-colon in the body of the caller function.

2. Function Prototype

It defines the structure of the function prior to the use of that function.

2.1 Define Function prototype

The first line of function declaration (i.e. the general syntax), which tells the compiler about the structure of the function, is known as function prototype. It mainly tells about the return type, function name and type of parameters that the function will be using during the function call. The general syntax of a function in Java is as follows –

```
<access specifier> <modifier> <return type> function_name (parameter list)
{
        Body of the function
}
```

where ,

i. **access specifier** – it tells in which part of the program, this function will be accessible. There are three main access specifiers:
   - public – a function with public access specifier can be access from anywhere within the class as well as from outside the class. It means the function has a public accessibility for everyone.
   - private – a function with private access specifier can only be accessed from the class where it is declared and not from anywhere else.
   - protected – a function with protected access specifier can only be accessed from the class where it is declared and from its sub classes (the classes those are inherited from that class) and not from anywhere else.
   - default - if a function is declared without any access specifier, then it is default in nature and it means the same as public but the function will not be accessible from any other class outside the package.

ii. **modifier** – it tells about the nature of the function i.e. it is a class member or instance member, or final type or non-final type function. Followings are the three modifiers for a function – static, final and synchronize. A function declared with **static** keyword means it is a class member whereas declared without **static** keyword means it is an instance member. Similarly as a function declared with **final** keyword means it is final in whereas declared without **final** keyword means it is non-final in nature.

iii. **return type –** it specifies the return type of the value that will be returned by the function when it is being called. It is any valid data type that type of value a function can return to it caller.

iv. **function_name** – it is the name given to the function using which the function will be referred within the program. It is any valid identifier name given to the function following the rules and conventions of naming an identifier.

v. **parameter list -** it specifies the type, number and sequence of parameters the function will have and according to which the function will receive its values from the caller during the time of function call.

2.2 Function Signature

The function signature is the list of parameters mentioned in the function prototype. It defines the type, number and sequence of parameters that will be received by the function from the caller function during the function call. Function signature plays an important role in function overloading.

2.3 Types of functions according to Function Prototype

We can divide our functions into the following four categories according to the function prototype:

- **void F (void)** – these types of functions neither receive any values from the caller nor return any value to the caller during the function.

```
public void add( ) // function declaration
{
    int a=5, b=6;
    int c=a+b;
    System.out.println("Sum="+c);
}
public void main( )
{
    add( ); // function call
}
```

- **void F (parameter)** – these types of functions receive some values from the caller but do not returns any value to the caller during the function.

```
public void add(int a, int b)       // function will receive the values of a and b
{                                   // during the function call
```

```
    int c=a+b;
    System.out.println(“Sum=”+c);
}
public void main( )
{
    int a=5, b=6;
    add(a,b); // need to pass the value at the function call statement
}
```

- **return_type F (parameter)** – these types of functions receive values from the caller and also return a value to the caller during the function.

```
public int add(int a, int b)        // function will receive the values of a and b
{                                   // during function call
    int c=a+b;
    return c;//the result will be returned back
}
public void main( )
{
    int a=5, b=6;
    int c=add(a,b); // need to pass the value at the function call statement
    System.out.println(“Sum=”+c);
}
```

- **return_type F (void)** – these types of functions does not receives any values from the caller but returns a value to the caller during the function.

```
public void add( ) // function declaration
{
    int a=5, b=6;
    return (a+b);
}
public void main( )
{
    int c=add( ); // function call
    System.out.println(“Sum=”+c);
}
```

3. Function definition

A function definition specifies the name of the function, the types and number of parameters it expects to receive, and its return type. A function definition also includes a function body with the declarations of its local variables, and the statements that determine what the function does.
Here is an example of a typical method declaration:

```
public double calculate Answer(double span, int nOfEngines, double length, double
grossTons)
{
   //do the calculation here
}
```

The only required elements of a method declaration are the method's return type, name, a pair of parentheses, (), and a body between braces, {}.

More generally, method declarations have six components, in order:

1. **Access Specifier** —such as public, private, and others.
2. **Modifier** —such as static, final and others
3. **The return type** —the data type of the value returned by the method, or void if the method does not return a value.
4. **The method name** —the rules for field names apply to method names as well, but the convention is a little different.
5. **The parameter list in parenthesis** — a comma-delimited list of input parameters, preceded by their data types, enclosed by parentheses, (). If there are no parameters, you must use empty parentheses.
6. **The method body, enclosed between braces** — the method's code, including the declaration of local variables, goes here.

4. Calling / Accessing a Function

A function is called or invoked by providing the function name followed by the parameters being sent enclosed in parenthesis in the function call statement of the caller function. The property of a function that states that any change inside a function can also be reflected outside is called side effect of the function.

4.1 Actual parameters and formal parameters

The parameters that appear in the function definition are formal parameters and those appear in the function call statement are actual parameters.
Formal parameters are the part of the function definition and actual parameters determine the actual values sent to the function.
Example program to show actual and formal parameters in the program-

```
//program to compute and display the sum of the given series :
//1!+2!+3!+…+10!,
//where a separate function is used for computing the factorial of each term.
class ExampleFact
{
        public int computeFactorial(int a) // here a is the formal parameter
        {
                int f=1;
                while(a>0)
                {
                  f=f*a;
                  a--;
                }
                return f;
        }
        public void computeSum( )
        {
                int sum=0, fact;
                for(int i=0; i<=10; i++)
                {
                  fact= computeFactorial(i); // here i is the actual parameter
                  sum += fact;
                }
                System.out.println("Sum of the series="+sum);
        }
    }
```

In the above program when the function computeFactorial( ) is called, variable i is passed to the function as actual parameter and in the definition of that function, variable a which is receiving the value of i is the formal parameter.

4.2 Call by Value (Pass by Value)

When a function is called by value, the formal parameters will receive duplicate values from their actual counter parts. In case of call by value or pass by value, there exist two copies of the same value in the memory as there is two separate memory locations for them.
Given below an example of function call using call-by-value method:

```
class Example
{
        public void change(int a)// called function, where a is the formal parameter
[a=10]
        {
                a++;
                System.out.println("Value of a="+a);
        }
        public static void main( )
        {
                int x=10;
                System.out.println("Before the function call, Value of x="+x);
                change(x);//function call, where x is the actual parameter
                System.out.println("After the function call, Value of x="+x);
        }
}
```

During the execution of the above program, we will see the following output –

Before the function call, Value of x=10
Value of a=11
After the function call, Value of x=10

We can see that there exist two separate variables x (for main method) and a (for change method) in the memory. Now during the function call to change, the variable a will receive a duplicate copy of the value 10 (stored in the variable x). So, when the after the execution of the function change( ), the value of a gets incremented to 11 but there will be no change to the value of the variable x.

Thus, we can conclude to the point that in the call-by-value method whatever changes made in the formal parameter will not be reflected back to the actual parameter.

4.3 Call by Reference (pass by Reference)

When a function is called by reference, the formal parameters will receive the address of the actual parameter instead of the duplicate values from them. In case of call by reference or pass by reference, thou there exist two separate memory locations in the form of two variable names, they share the same value.
Given below an example of function call using call-by-value method:

```
class Example
{
        int a; // instance variable
        public void change(Example ob1) //function will receive object’s reference
        {
                ob1.a++; // ob1.a = ob2.a => a++; (the data member of the class)
                System.out.println(“Value of a in change=”+ob1.a);
        }
        public static void main( )
        {
                Example ob2=new Example( );
                ob2.a=10;
                System.out.println(“Value of a before function call =”+ob2.a);
                ob2.change(ob2); // passing an object
                System.out.println(“Value of a after function call =”+ob2.a);
        }
}
```

During the execution of the above program, we will see the following output –

Value of a in main, before function call =10
Value of a in change=11
Value of a in main, after function call =11

We can see that there exist two separate object ob2 (for main method) and ob1 (for change method) in the memory. Now when the function change() is called from the main(), ob1.a will receive the address of ob2.a instead of the value. So, after the execution of the function change(), when the value of ob1.a gets incremented to 11 the same will be reflected back to ob2.a also.

Thus, we can conclude to the point that in the call-by-reference method whatever changes made in the formal parameters will reflect back in the actual parameters.

In general, whenever any object or array or String is passed to a function, it is called by reference and for all other type of variables it is call be value.

5. Returning a value from a Function (Return statement in the function)

A function is terminated when there is a return statement encountered or the last statement in the function is executed. In case of return statement, a value is returned back to the caller function if the return type of that function has a valid data type. Generally, a return statement is used to terminate a function whether or not it returns a value.

The return statement provides the following two uses in the function:
1. Immediate exit from the function and passing the control back to the caller.
2. It is used to return a value to the calling code.

A function can return only one value at a time with the help of return statement. A function may contain more than one return statements but only one of them gets executed because the execution of the function terminates as soon as a return statement is encounter.

6. Getter and Setter methods in Classes

Generally while declaring a class structure, we keep all the member variables as private and all the member functions as public and thus only the member functions can access these variables from inside the class. This gives us a complete control over the member variables of the class.

But some time it is needed to access the member variables or the values stored in them from outside the class. For this purpose we have the following two types of functions –
1. Getter method or Accessor method
2. Setter method or Mutator method

6.1 Getter Method/Accessor Method

A getter method or accessor method is a function of a class that returns the value of a data member of that class to the outsider. Accessor methods are provided for private members of a class. One accessor method can return only one data member's value.

6.2 Setter Method/Mutator Method

A setter method or mutator method is a function of the class that sets or changes the value of a data member of that class. Mutator methods are provided for private members of a class. One mutator method can assign the value(s) of one or all data member(s).

6.3 Pure and Impure functions

A function that changes the state of the object, whose method is being called, is known as pure method and a function that does not changes the state of the object is known as impure method.

7. Static Functions in Classes

Any function declared with **static** keyword in the class is a class method or static method of that class. The property of a class method or static method is that there exists only one copy of the method in the memory and all the object of that class shares the same copy of that method. Another property of these methods are, if any change is made in the values of variables of a static method by any one function call (may be through same object or through more than one object) will be reflected in the next function call.

8. The main( ) Method

main( ) method is a special method from where execution of the program begins. For this reason, main( ) method is a compulsory method of a class. It is the main entrance to the program from the outside world.
It is a user defined method that holds the entire program coding during the execution of the program. In Java programming, the syntax of main( ) method is as follows:

```
public static void main(String args[ ])
{
  …..
  …..
}
```

From the above syntax, we can see that main( ) method is publicly accessible, static in nature, has no return type. It is a class member method and hence only one copy of main exists in the memory. The parameter – String args[] signifies that it is a command line argument that can receive an array of String type as input from the user during the execution of the program.

A sample program showing command line input is given below:

```
//program to compute simple interest and add that to the principle amount, where
principle, //rate and time will be entered by the user from the command line using
command line arguments.
class ExampleSI
{
        double prc;
        double rate;
        int time;
        public ExampleSI(double p, double r, int t)
        {
                prc=p;
                rate=r;
                time=t;
        }
        public void computeSI( )
        {
                double si=(prc*rate*time)/100;
                prc+=si;
                System.out.println("Amount="+prc);
        }
        public static void main(String args[])
        {
                double p=Double.parseDouble(args[0]);
                double r=Double.parseDouble(args[1]);
                int t=Integer.parseInt(args[2]);
                ExampleSI obj=new ExampleSI(p,r,t);
                Obj.computeSI();
        }
}
```

In the above program, String args[] is the command line argument which hold the inputs (principle amount, rate and time) from the user and those will be passed to the variables accordingly. Actually agrs[] is a String array which holds the values separated by space in String format at different index positions as args[0], args[1], args[2],….so on. They are then required to parsed into the respective data type to store them in the corresponding variables.

9. Constructor (Special method of a Class)

Constructor is a special member method of a class that has the same name as the class. It is used to initialise the data members of a class object at the time of object instantiation. The process of initialising the instance variables of an object at the time of memory allocation is known as object instantiation.

9.1 Properties of a constructor

A constructor of a class has the following properties:-
1. It has the same name of the class.
2. It has no return type, not even void.
3. It must be a public member of a class.
4. A constructor cannot be called explicitly unlike other member methods of a class. It is called implicitly at the time of object creation using **new** keyword.
5. A constructor is mainly used to initialize the data members (instance variables) of a class at the time of object creation.
6. A class can have multiple constructor, but in that case each constructor must be different from other one in its function signature. (This concept of multiple constructor in a class is known as constructor overloading).

In the above example, the method with the name of the class is the constructor of that class (**ExampleSI(double, double, int)**) and the statement - **ExampleSI obj=new ExampleSI(p,r,t);** where the object of the class, obj is created using new keyword, the constructor of the class gets invoked implicitly.

9.2 Type of constructors

We can use two types of constructors in a class – **parameterized** and **non-parameterized**.

1. **Parameterized constructor** is a constructor that has parameters in its function definition and initializes the data members with the values passed by those parameters. In the above example, **ExampleSI(double p, double r, int i)** is a parameterized constructor that is initializing the data members prc, rate and time with the values from p, r, t respectively.

2. **Non-parameterized constructor** is a constructor that does not have any parameters in its function definition and this type of constructor generally initializes the data members with either their default values or with some fixed set of values. And each time an object is created using non-parameterized constructor, the data members of that object gets a same set of values there.

9.3 Default Constructor

A non-parameterized constructor is said to be a default constructor when it initializes the data members of the class with their default values.

```
//Example of a non-parameterized constructor acting as a default constructor
class ExampleStudent
{
        //data members
        int roll;
        String name;
        double marks;
        char grade;

        public ExampleStudent( ) //non-parameterized constructor
        {
                roll=0;
                name= “”;
                marks=0.0;
                grade= ‘\u0000’;
        }
        …….
        …….
}
```

In the above program code, the non-parameterized is initializing the data members with their default values. (0 for int, 0.0 for double, “” for String and ‘\u0000’ for char data type)
Also if a programmer does not declare any constructor in the class, then at the time of object creation, the compiler calls the default constructor (of the System) to initialize the data members with their default values.

9.4 Constructor Overloading

Like function overloading, we can have constructor overloading in a class i.e. there can be more than one constructor having a different set of parameters.
Given below is an example program on constructor overloading

```
class ExampleTime
{
        int hour, minute, second;
        public ExampleTime( ) // non-parameterized constructor
        {
                hour=1;
                minute=1;
                second=1;
        }
        public ExampleTime(int h, int m) // constructor with 2 arguments
        {
                hour=h;
                minute=m;
                second=0;
        }
        public ExampleTime(int h, int m, int t) // constructor with 3 arguments
        {
                hour=h;
                minute=m;
                second=s;
        }
        …..
        …..
}
```

9.5 Copy Constructor

Copy constructor is a special type of parameterized constructor that has an object of the same class as its parameter. Copy constructor is mainly used in the program to create an object which will initialize its data members with the values passed from another object's data members.

```
//Example program on copy constructor
class ExampleTime

{
        int hour, minute, second;
        public ExampleTime(int h, int m, int t) // parameterized constructor
        {
                hour=h;
                minute=m;
                second=s;
        }
        public ExampleTime(ExampleTime T) // copy constructor with an object
        {                                            // (T) as parameter
                hour=T.hour;
                minute=T.minute;
                second=T.second;
        }
        …..
        …..
        public static void main(String ar[])
        {
                ExampleTime obj1=new ExampleTime(12,45,56);
                ExampleTime obj2=new ExampleTime(obj1);
                ExampleTime obj3=new ExampleTime(0,0,0);
                obj3=obj1;
                …..
                …..
        }
}
```

9.6 Difference between Constructor and function

Constructor	**Function/Method**
It creates an instance of a class	It stores a group of statements which are executed to perform a specific task
It has the same name of the class and has no return type, not even void	It must be with a separate name from the class and must have a valid return type or void
Constructor cannot be called explicitly	Need to be called in the caller method using call statement
Constructor cannot be inherited by the child class from the parent class	Methods are inherited to the child class from its parent class.
Constructor cannot be recursive.	Recursive function is possible

9.7 Similarity between Constructor and function

1. Both can be parameterized or non-parameterized
2. We can perform any computation in both of them.
3. Both can be overloaded i.e. a class can contain more than one constructor like more than one function with same name but the signature should be different.

10. The "this" variable

As we know that only one copy of member functions is maintained that is shared by all the objects of the class and on the other hand every objects have a separate copy of their data members, there might be a situation when an object is called a function and there are three objects in memory. How does the member function decide which object's data member it should work upon?
Answer to it is **this** keyword. When a member function is called, it is automatically passed an implicit argument that is a reference to the object that invoked the function. This reference is called '**this'.**

this keyword returns the reference of the current object whose method is being invoked. Thus which ever object is currently doing a function call, this pointer points to the object.

Points to remember:
The keyword **this** pertains to an instance method and not to a class methods
The keyword **this** is an implicit argument to the current object of a class
The keyword **this** is accessible inside of any instance method.

Solved questions (on Functions/Methods)

1. Define Function. What is function prototype?

Ans. A method or function is a sequence of statements that carry out specific task.

The first line of the function definition is function prototype that tells about the type of value returned by the function and the number and type of arguments.

2. What are the actual and formal parameters of the function?

Ans. **Actual parameters** are the parameters appearing in the function call statement.

Formal parameters are the ones that appear in function definition.

3. What is the statement specifically called that invokes a function?

Ans. Function call or Method call statement

4. How many values can be returned by the function?

Ans. One

5. What is the condition of using a function in an expression?

Ans. Only the function returning a value can be used in expression.

6. When a function returns a value, the entire function call can be assigned to a variable. T/F?

Ans. True.

7. Identify the errors in the function below:

Ans.
a) float average(a, b) { } ------ float average(int a , int b) { }
b) float mult(int x, y) { } ------ float mult (int x, int y) { }
c) float doer (int, float = 3.14) { } ------ float doer (int x, float y) { }

8. Given the function below write a main function that includes everything necessary to call this function.

```
int thrice ( int x)
{
    return a*3;
}
```

Ans.

```
class sample
{
        int thrice( int a)
        {
                return a*3;
        }

        public static void main( )
        {
                int result= thrice(2);
                System.out.println( result);
        }
}
```

9. What is the principal reason for passing arguments by value?

Ans. The call by value method copies the values of actual parameters into formal parameters, that is the function creates its own copy of argument values and then uses them.

10. What is the principal reason for passing arguments by reference? In a function call, what all data items can be passed by reference?

Ans. When a function is called by reference, then the formal parameters become references to the actual parameters in the calling function. That is, the called function does not create its own copy of original values; rather it refers to the original value only by different names.

11. What is the role of the return statement in the function?

Ans. The return statement is useful in two ways. First an immediate exit from the function is caused as soon as the return statement is encountered. Secondly it is used to return a value to the calling code.

11. What are three types of function in Java?

Ans. Computational, Manipulative, Procedural

12. Write a function that interchanges the value of two integers A and B without using any third variable.

Ans.
```
public void exchange( int a, int b)
{
        a= a+b;
        b=a-b;
        a= a-b;
        System.out.println(a +" and" +b);
}
```

13. Differentiate between CALL by reference and CALL by value.

Ans. In **call by value**, the called function creates its own work copy for the passed parameters and copies the passed value in it. Any changes that take place remain in the work copy and the original data remains intact.

In **call by reference**, the called function receives the reference to the passed parameters and through this reference, it accesses the original data. Any changes that take place are reflected in the original data.

14. What is polymorphism? How does function overloading implement polymorphism?

Ans. Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to
Refer to a child class object.
A function name having several definitions in the same scope that are differentiable by the number or types of their arguments, is said to be an overloading function, thus showing polymorphism.

15. What is function overloading?

Ans. A function name having several definitions in the same scope that are differentiable by the number or types of their arguments, is said to be an overloading function.

16. What is the significance of function overloading in Java?

Ans. Function overloading not only implements polymorphism but also reduces the number of comparisons in a program and thereby makes the program run faster.
Objects have characteristics and behaviour. The same behaviour of the object may differ in different situations. So it can happen with functions also. After all it is class that implements OOP in practice. Therefore in order to simulate real world objects in programming , it is necessary to have function overloading.

17. What is the role of a function's signature in disambiguation process?

Ans. To overload a function declare and define all the function with the same name but different signatures. The signature can differ in the number of arguments or in the type of arguments, or both.

18. What factors make two definitions with the same function name significantly different?

Ans. The number and the type of the argument.

19. How does the use of constant suffixes help avoid ambiguity when an overloaded function is called?

Ans. To avoid ambiguity, use of constant suffixes (F, L, D) distinguishes between values as these greatly help in indicating which overloaded function should be called.

Solved questions (on Constructor)

1. What is a Construction? What does it do?

Ans. Constructor is a member function that has the same name of its class. It initializes the data members of class-object with legal initial values.

2. What is that class called which does not have a public constructor?

Ans. Private class.

3. A constructor is executed when an object is created.

4. Write a class specifier (along with its constructor) that creates a class student having two private data members : rollno and grade and two public functions init( ) and display( ).

Ans.

```
class Student
{       private int  rollno;
        private char  grade;
        public Student( int r, char g)
        {       rollno = r;
                grade= g;
        }
        public void init( )
        {
        }
        public void display( )
        {
        }
}
```

5. Can you think of the benefits of a private constructor if any? What are they?

Ans. A private constructor is not available to the non member function. That is an object of the same class cannot be created in a non member function .

6. Here is a skeleton definition of a class:

```
class Sample {
                        int i; char c; float f;
                        :
                }
```

Implement the constructor.

Ans.

```
        class Sample
        {
                int i;
                char c;
                float f;
                public Sample( int in, char ch, float fl)
                {
                        i= in;
                        c= ch;
                        f= fl;
                }
        }
```

7. Define a constructor function for a **Date class** that initializes the Date object with given initial values. In case initial values are not provided , it should initialize the object with the default values.

Ans.

```
class Date
{
        int dd,mm yy;
        public Date ( )
        {   dd=0;
                mm=0;
                yy=0;
        }
        public Date( int d, int m, int y)
        {
                dd=0;
                mm=0;
                yy=0;
        }
        public void useDate( )
        {
                Date date = new Date (1, 1, 2013);

                Date date1= new Date( );
        }
}
```

8. What condition a function must specify in order to create object of a class?

Ans. A function having the same name as the name of the class and no return type , not even void.

9. Constructor function obeys the usual access rules. What does this statement means?

Ans. This statement means a private or protected constructor is not available to the non member functions. In other words with a private or protected constructor we cannot create an object of the same class in non-member function, however this is allowed in the member functions .

10. How are parameterizes constructors different from non-parameterized constructors?

Ans. A constructor receiving arguments is called as parameterizes constructor whereas a constructor taking no arguments is non-parameterized constructor.

11. What are the benefits/drawbacks of temporary instances?

Ans. A temporary instance remains in the memory as long as the statement defining it is getting executed , after the statement , this object is automatically destroyed and memory is released. Therefore , memory remains occupied only for the time when it is needed.

12. How do we invoke a constructor?

Ans. A constructor is invoked when an object is created.

13. How can objects be initialized with desired values at the time of object creation?

Ans. By using parameterized constructor.

14. When a compiler can automatically generate a constructor if it is not defined then why it is said that it is writing constructor for a class is a good practice?

Ans. Because the default constructor provided by the compiler does not do anything specific . It initializes the data members by any dummy value.

15. 'Accessibility of a constructor greatly affects the scope and visibility of their class'. Elaborate this statement.

Ans. This statement means a private or protected constructor is not available to the non member functions. In other words with a private or protected constructor we cannot create an object of the same class in non-member function, however this is allowed in the member functions . Consider the example,

```
class X
{       int i;
        private X( ) {
        i=10; j= 10;    k=10;   }
        public int j, k;

        public void getval(  )
        {       :       }
        void check (  )
        {       X obj =         new X( );
                :       }

}
```

```
class Y
{
        public void test ( )
        {       X obj =         new X( );
                        :
        }
}
```

In the above example, since constructor of X is private, objects of X can be created only inside the member function, but not inside non-member functions. Reason being that every time an object is created the constructor is automatically invoked; but if the function declaring the object, does not have access privilege for the constructor, it cannot be invoked for the object, thus the object cannot be created under such function.

16. List some of the special properties of the constructor functions.

Ans. Some of the properties of constructor are as follows:

(i) It has the same name of that of the class where it has been declared.

(ii) It has no return type not even void

(iii) It cannot be called explicitly, only called implicitly at the time of object creation

(iv) Constructors can be overloaded

17. What is parameterized constructor? How it is useful?

Ans. A constructor that receives arguments is called Parameterized constructor. It allows us to initialize the various data elements of different objects with different values when they are created. This is achieved by passing different values as arguments to the constructor function when the objects are created.

18. Design a class to represent a Bank account. Include the following members:

Data members

a) Name of the depositor	b) Type of the account
c) Account number	d) Balance amount in the account

Methods

a) To assign initial values	b) To deposit an amount
b) To withdraw an amount	d) to display the name and balance
e) do write the proper constructor functions.	

Ans.

```
class BankAccount
{
        private String DepositorName ;
        private long AccountNumber ;
        private String AccountType ;
        private double BalanceAmount ;
```

```
public BankAccount( )//default constructor
{
     DepositName = " " ;
     AccountNumber = 0;
      AccountType = " " ;
     BalanceAmount = -1 ;
}
public BankAccount (String dName, long accno, String accType,
                                        double balAmount)
{
     DepositorName = dName;
     AccountNumber = accno;
       AccountType = accType ;
     BalanceAmount = balAmount ;
}
public void initialize(String dname, long accno, String accType,
                                        double balAmount )
{
     DepositorName = dName;
     AccountNumber = accno; AccountType = accType ;
     BalanceAmount = balAmount ;
}
public void display ( )
{
     System.out.println ("Depositor Name :" + DepositorName) ;
     System.out.println ("Account Name :" + AccountNumber) ;
     System.out.println ("Account Type :" + AccountType) ;
     System.out.println ("Balance Amount :" + BalanceAmount) ;
}
public void deposit (double amount)
{
     BalanceAmount += amount;
}
public void withdraw (double amount)
{
     if (amount <= BalanceAmount)
       BalanceAmount -= amount;
}
public static void main (String args[ ])
{
     BankAccount acc1 = new BankAccount ( );
     Acc1.initialize ("Chetan", 31290, "Saving", 8000) ;
     BanAccount acc2 = new BankAccount ("Ronald", 41777, "current",
     70000) ;
```

```
                acc1.deposit(17000) ;
                acc1.display( ) ;
                acc2.withdraw(20000) ;
                acc2.display( ) ;
            }
}
```

Unsolved questions

Q1. What is a function? Write down the complete syntax of a function in Java

Q2. What are function parameters? Differentiate between actual and formal parameters.

Q3 . Identify the errors on the following function skeletors given below:

(i) float average(a,b) { }

(ii) int mult(int x, y, z) { }

(iii) void display( ){ return 1; }

(iv) void check(int a, int b) System.out.println(a+b);

Q4. When an argument does is passed by reference? What is the difference between passing by value and passing by reference?

Q5. What is function overloading? Explain with an example program in Java.

Q6. What is the significance of return keyword in a function in Java?

Q7. What is the output of the following code?

```
void function( String s )
{
        String s = “Java Programming”;
        String s1 = “xyz”;
        System.out.println(“s + s1=”  (s+s1));
}
```

Q8. Complete the following program code that will compute and display the all the factors of a number passed as a parameter to the function given below:

```
void factor( _____ )
{
        int i=1, f=0;
        while( ________ )
        {       if(n % i = = 0 )
                System.out.println( _____ );
                _______
        }
}
```

Q9. What is the significance of constructor in Java program?

Q10. What is the class called as if that does not have a public constructor? Why it is called so?

Q11. How parameterized constructor is differs from non-parameterized constructor?

Q12. What are temporary instances? How they are different from normal objects of a class?

Q13. What is the significance of 'this' keyword?

Q14. Explain constructor overloading with an example program.

Q15. Declare a class named 'Time' with data members to store hour, minute and second separately and one non-parameterized constructor to set the time as 0hr 0 min 0 sec, one parameterized constructor to initialize the data members through parameters.

Q16. Declare a class to represent Student to store roll no., name and grade and initialize the data members using parameterized constructor having the same argument names in the argument list.

Multiple Choice questions on Methods

1. What is the return type of a method that does not returns any value?

a) int

b) float

c) void

d) double

Answer: c

Explanation: Return type of an method must be made void if it is not returning any value.

2. What is the process of defining more than one method in a class differentiated by method signature?

a) Function overriding

b) Function overloading

c) Function doubling

d) None of the mentioned

Answer: b

Explanation: Function overloading is a process of defining more than one method in a class with same name differentiated by function signature i:e return type or parameters type and number. Example – int volume(int length, int width) & int volume(int length , int width , int height) can be used to calculate volume.

3. Which of the following is a method having same name as that of it's class?
a) finalize
b) delete
c) class
d) constructor
Answer: d
Explanation: A constructor is a method that initializes an object immediately upon creation. It has the same name as that of class in which it resides.

4. Which method can be defined only once in a program?
a) main method
b) finalize method
c) static method
d) private method
Answer: a
Explanation: main() method can be defined only once in a program. Program execution begins from the main() method by java's run time system.

5. Which of these statement is incorrect?
a) All object of a class are allotted memory for the all the variables defined in the class.
b) If a function is defined public it can be accessed by object of other class by inheritance.
c) main() method must be made public.
d) All object of a class are allotted memory for the methods defined in the class.
Answer: d
Explanation: All object of class share a single copy of methods defined in a class, Methods are allotted memory only once. All the objects of the class have access to methods of that class are allotted memory only for the variables not for the methods.

6. What is the output of this program?

```
class equality {
    int x;
    int y;
    boolean isequal(){
        return(x == y);
    }
}
class Output {
   public static void main(String args[])
   {
      equality obj = new equality();
      obj.x = 5;
      obj.y = 5;
      System.out.println(obj.isequal());
   }
}
```

a) false

b) true

c) 0

d) 1

Answer: b

7. What is the output of this program?

```
class Output {
   static void main(String args[])
   {
      int x , y = 1;
      x = 10;
      if (x != 10 && x / 0 == 0)
         System.out.println(y);
      else
         System.out.println(++y);
   }
}
```

a) 1

b) 2

c) Runtime Error

d) Compilation Error

Answer: d

Explanation: main() method must be made public. Without main() being public java run time system will not be able to access main() and will not be able to execute the code.

Multiple Choice questions on Class & Constructor

1. classes are useful because they
A. permit data to be hidden from other classes
B. can closely model objects in the real world
C. brings together all aspects of an entity in one place
D. all of the above.

2. Which of the following is the correct statement to create an object of Data class?
A. Data d=new object(); B. Data d=new Data();
C. Data d()=new Data(); D. Data d()=new Data();

3. The new keyword is used to ____
A. call a method of a class
B. Allocate memory to an object
C. Release memory of an object
D. none of above

4. A constructor is a special type of_____
A. class B. variable C. method D. object

5. A default constructor ______
A. has no return type B. has no argument
C. has one argument D. has one argument but no return type.

6. To specify default access to a variable or a method ___ keyword is used
A. public B. private C. default D. none of above

7. The .dot operator connects the following two entities :
A. a class member and a class object B. a class object and a class
C. a class and a member of that class D. a class object and a member of that class

8. When a variable is declared as static , _____
A. It is automatically initialized before an object of its class is created.
B. It becomes constant
C. Its value is changed every time an object of its class is created.
D. It becomes public.

9. Before doing garbage collection , _____ method is called :
A. main() B. finalize() C. final() D. collect()

10. Inheritance means______
A. ability to take more than one form B. data hiding
C. ability to use properties of another class D. wrapping up of data and methods

11.To inherit from class ___ keyword is used.
A. inherit B. extends C. uses D. implements

12.Java does not support ____ inheritance
A. multilevel B. multiple C. Hierarchical D. Single

13. When method is overridden , then by subclass object which class 's method is called

A. super class B. subclass C. both D. none

14. When class is declared as abstract , then ____

A. Its object cannot be created B. Its subclass cannot be created
C. It cannot inherit any class D. It cannot have methods

15. When a class is declared as final , then ____

A. It cannot be inherited B. It must be inherited
C. Its object cannot be created D. none of these

16.___ keyword is used to refer to the current object

A. super B. this
C. new D. volatile

17. ____ are automatically called when an object is destroyed

A. collectGarbage() B. Destructor()
C. finalize() D. final()

18. Overloaded methods ____

A. are a group of methods with the same name
B. have the same number and type of arguments
C. make life simpler for programmer
D. may fail unexpectedly due to stress.

19.A recursion occurs when ___

A. a constructor calls a method B. A method calls itself
C. a method calls another method D. A constructor calls another constructor.

20. super keyword can be used to _____

A. call super class ‘s constructor
B. access super class ‘s member
C. both a and b
D. none of the above

Answers:

1	**- D**	2	**- B**	3	**- B**	4	**- C**	5	**- B**
6	**- D**	7	**- D**	8	**- A**	9	**- B**	10	**- C**
11	**- B**	12	**- B**	13	**- B**	14	**- B**	15	**- A**
16	**- B**	17	**- C**	18	**- A**	19	**- B**	20	**- C**

Short answer questions:

1. What are functions? Write and explain the complete syntax of the function.
2. Explain pure and impure functions in Java using an example program.
3. Differentiate between call by value and call by reference.
4. Differentiate between static and non-static methods of a class.
5. Differentiate between formal and actual parameters with an example program.
6. What is the significance of main( ) method in a Java program?
7. What is the significance of **‘String args[]’** in main method?
8. Give the output of the following code snippets showing use of various methods:
    a.
```
int dispF(int x, int y)
{
        if( x>=y)
        {       x = x – y;
                return dispF(x, y);
        }
        else
                return x;
}
```
    b.
```
void example(int n)   //if n=11
{
        if( n% 2 == 0)
                return 1;
        else
                return function( n/2 )* 10 + n%2;
}
```
    c.
```
double Disp(double x, int n)
{
        int inv = 0;
        double res = 1.0;
        if(n= =0)
                return 1;
        else if( n<0 )
        {       inv = 1;
                n = n * (-1);
        }
        for( int i=1; i <= n; i++)
                res *= x;
        if( inv = =1)
                res = 1.0/res;
        return res;
}
```

d. **boolean unknown(int n)   //if n=10**

```
{
        int i, k, s;
        for(k=n;k>9; )
        {
                for(s=0, i = k; i!= 0; i= i/10)
                        s = s+ i %10;
                k = s;
        }
        boolean b = (s= =1) ? true : false;
}
```

e. **void series(int n)   // if n=5**

```
{       if(n<=0)
                return 1;
        else
         {      sum+=((n+1)*(n+2)*(n+3));
                series(n-2);
        }
}
```

Programs on Functions and Constructor:

Question 1

A class Number has been defined to find the frequency of each digit present in it and the sum of the digit and to display the results. Some of the members of the class Number are given below:

Class name	Number
Data member	num – long integer type
Member functions:	
Number( )	constructor to assign 0 to num
Number(long a)	constructor to assign a to num
void digitFrequency( )	to find the frequency of each digit and to display it.
int sumDigits( )	to returns the sum of the digits of the number.

Specify the class Number giving the details of the two constructors and functions void digitFrequency( ) and int sumDigits( ). You do not need to write the main function.

Question 2

Consider N-digit number K. Now square it and add the right n-digit to the left n or n-1 digit. If the resultant sum is K then it is a Kaprikar number. e.g. 297 is a **Kaprikar number**

Working: $(297)^2 = 88209$
$88+209 = 297$

Create a class name **KNumber** with following class description:

Data Members:- m, n (two integers where m < n)

Member function:-

void input( )	this function will accept two integers and store them in m and n.
long power(int z)	this function will return the square of z
void checkKaprikar(int z)	this function will check the number z is a Kaprikar number or not. If yes, then the function will display the working along with the number as shown in the example above.
void display( )	this function will display all the Kaprikar numbers between m & n

WAP to declare the above class with its member functions. No need to do the main function.

Question 3

A class called OddSeries has been defined to find the smallest value of integer n, such that,

$3+\frac{9}{2!}+\frac{27}{3!}+.....+\frac{3^n}{n!}\geq S$, where S is any positive floating point value between 6 and 7

Declare a **class OddSeries** with following class description:-

Data members:

long n	long integer variable to store number of terms
float S	float variable
float k	float variable to store the value of series evaluated.

Member functions:-

OddSeries( )	default constructor
void accept( )	to accept the value of S, where $6\leq S\leq 7$
void display( )	calculates and display the least value of n.
long fact(long x)	to compute and return factorial of x

(a) WAP to declare the above class with its member functions. No need to do the main function.

(b) What is the difference between signed int and unsigned int.

Question 4

Adding the two previous terms of the series **generates Fibbonacci numbers**. e.g., if we start with 1 and 1, then the series will be –

1, 1, 2, 3, 5, 8, 13, 21, ……….

A natural number is said to be prime if it has exactly 2 divisors, i.e. 1 and itself. e.g., 2, 3, 5, 7, 11, …

Define a function **isPrime ( )** with the following declaration :-

int isPrime (int), which

returns 1 if the argument of the function is prime (e.g. 3)

returns 0 if the argument of the function is composite (e.g. 4)

returns 1 if the argument of the function neither prime nor composite (e.g.1)

Design a program in Java to use this function isPrime ( ) to output all the Fibbonaci numbers that are prime in the range 1 to 1000.

Question 5
Design a program in Java to enter two 4-digit numbers, then print all the numbers between them with following properties in a menu driven format:-

- It's first two digits are equal
- It's last two digits are equal
- It's a perfect square.

Question 6
A fractional number is consisting of two parts – numerator and denominator. Addition of two fractional numbers is done in the following way. First get the LCM of the two denominator parts that is the resultant denominator. Now divide that with each of the denominators and multiple the results with corresponding numerator. Now add the two results to get the resultant numerator.
Declare a class named "**Fraction"** with following description:-
Data members to store **numerator and denominator** of three fractional numbers (two fractional numbers as input from the user and one for storing the resultant fractional number of the addition).
Member function:-

void input( )	this function will ask the user to enter to fractional numbers in numerator and denominator.
int findLCM(int, int)	this function will return the LCM of two denominators passed as two parameters to the function
void addFraction( )	this function will do the addition process upon those two fractional numbers entered by the user and store the result in appropriate data members.
void display( )	this function will display all the three fractional numbers.

(a) WAP to declare the above class with its member functions. No need to do the main function.
(b) Give one advantage and one disadvantage of modularization in a program.

Question 7
A prime number is a number that is divisible by 1 and that number. Twin prime numbers are the pair of 2 prime numbers whose difference is 2, e.g (3,5), (5,7), (11,13) etc. The sum of reciprocals of the twin primes converges to a sum, known as **Brun's Constant** i.e.,

$$\left(\frac{1}{3}+\frac{1}{5}\right)+\left(\frac{1}{5}+\frac{1}{7}\right)+\left(\frac{1}{11}+\frac{1}{13}\right)+\text{.........upto } n^{th} \text{ term}$$

Declare a class named "**Primes"** with one data member **double sum** and three member functions **int primeCheck(int)**, **void TwinPrime(int, int)** and **void BrunConstant (int)**.
(a) WAP to declare the above class with its member functions. Use recursive technique in primeCheck( ) function. The main function need not be written.
(b) Give one advantage and one disadvantage of recursion over iteration.

Question 8

Adding the two previous terms of the series **generates the next Fibonacci numbers**. e.g. if we start with 1 and 1, then the series will be – 1,1,2,3,5,8,13,21,………

Declare a class **FiboSeries** with following class description:-

Data members	**-**	**integer variable to assign size, one integer array**
Member functions	**-**	
FiboSeries(int N)	-	to assign the size
int prime(int)	-	to check for prime and returns 1 or 0 accordingly
void fibouptoN( )	-	generate all the Fibonacci numbers upto N terms
void fiboNterm( )	-	generate all the Fibonacci numbers from 1 to N
void nonFibo( )	-	generate all the non-fibonacci numbers upto N terms
void primeFibo( )	-	generate all the prime fibonacci numbers upto N terms
void menu( )	-	that runs a menu to call the functions as per user's choice.