Chapter 4: Primitive Values and Types

The topics

1. JAVA Character Set

The character set is a set of valid characters that a language can recognize. A character represents any letter, digit or any other character entered from the keyboard.

Java uses the Unicode character set. Unicode is a two-byte character code set that has characters representing all the characters in almost all human alphabets, which includes English, Hindi, Arabic, Bengali, Chinese etc. Unicode characters can be referred to by using the escapes sequence \u followed by a four-digit hexadecimal number.

What does a character set contain?

- Alphabets – A to Z, a to z
- Digits – 0 to 9
- Special characters - *, & , !, …
- White space – ' ' (single space), tab, enter, new line…

ASCII values of character set:
A-Z : 65 to 90
a-z : 97 to 122
0-9 : 48 to 57
Space : 32
Special chars. 33 to 47

2. Tokens

The smallest individual unit in a program. Tokens are the building block of any programming language at the lowest level. There are five sets of tokens. They are as follows:-

2.1 Keywords

These are the words that **convey a special meaning to the language compiler**. These are reserved for special purposes & must not be used as normal identifier names.

Some of the KEYWORDS:- auto, break, byte, case, default, do, double, else, float, for, if, int, import, include, long, return, static, switch, while, class, void, static, public.

2.2 Identifiers

These are fundamental building blocks of a program & used as the names for variables, arrays, functions, objects, classes, and structures. Following are the Identifier forming rules:

- Identifier can have alphabets, digits & underscore.
- They must not begin with a digit.
- **They must not be a keyboard.**
- **They must not have space in between.**
- They are case-sensitive.

2.3 Literals

These are often referred as constants, & are data items that are fixed data values. There are five types of LITERALS:-

- INTERGER LITERALS – whole nos. without any fractional part.
- FLOATING LITERALS – numbers having fractional part.
- CHARACTER LITERALS – one character enclosed in single quotes.
- STRING LITERALS – multiple character enclosed in double quotes.
- NULL LITERALS - ' \0', 'null', its ASCII value is 0.
- BOOLEAN LITERALS - true and false (reversed words)

2.4 Punctuators

- Parenthesis ( ) – to denotes function
- Braces { } – start &end of a block
- Bracket [ ] – represent array
- Semicolon (;) - end of an executable statement
- Comma (,) - separate arguments, variable etc.

2.5 Operators

These are symbols that trigger some action on operands.

Operands are the variables, constants upon which operator operates.

Operator can be classified into 3 groups -

- UNARY OPERATOR, that acts on a single operand. e.g. +, - , ++, --.
- BINARY OPERATOR, that acts on two operands. e.g. 2+3, a = 4, 6>3.
- TERNARY OPERATOR, that acts on three operands. (?:)

WAP to assign length and breadth of a rectangle and compute area and perimeter. Display them.

```
class Rectangle
{
        public static void main()
        {
                int ln = 20, br = 12;
                int ar = ln * br;
                System.out.println("Area ="+ar);
                int pr = 2 * (ln + br);
                System.out.println("Perimeter="+pr);
        }
}
```

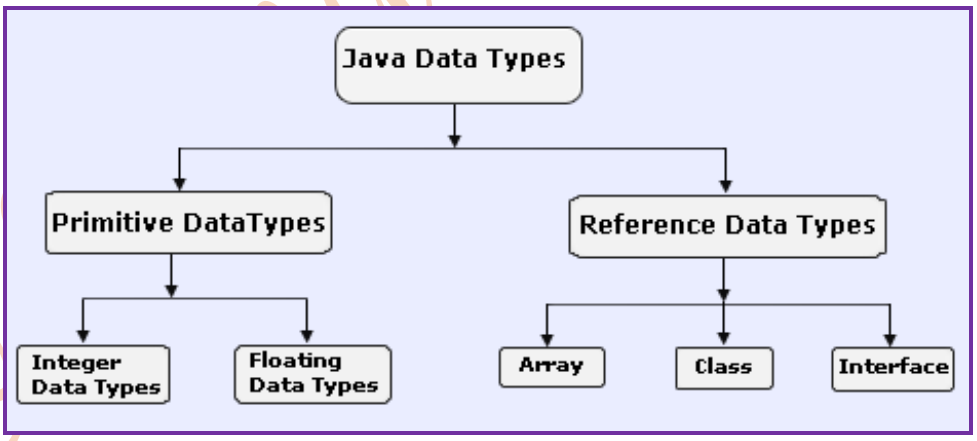
Keywords	class, public, static, void, main, int, System
Identifiers	Rectangle, ln, br, ar, pr
Literals	20, 12, "Area=", "Perimeter="
Operators	=, +, *
Punctuators	{ }, ( ), , ;

3. Data types in JAVA

3.1 What are Data types?

The data type defines a set of permitted values on which the legal operations can be performed.
In Java, all the variables need to be declared first i.e. before using a particular variable, it must be declared in the program for the memory allocation process. Like **int pedal = 1;** in this statement exists a field named **"pedal"** with the numerical value as **1**. The value assigned to a variable determines its **data type,** on which the legal operations of Java are performed.

The data types in the Java programming language are divided into two categories and can be explained using the following hierarchy structure :



3.2 Primitive Data Types

The primitive data types are **predefined** data types, which always hold the value of the same data type, and the values of a primitive data type don't share the **state** with other primitive values. These data types are named by a **reserved keyword** in Java programming language.

There are **eight primitive data types** supported by Java programming language :

i. **boolean**
   1-bit. May take on the values true and false only.
   true and false are defined constants of the language and are not the same as True and False, TRUE and FALSE, zero and nonzero, 1 and 0 or any other numeric value. Booleans may not be cast into any other type of variable nor may any other variable be cast into a boolean. (true and false are reserved words, but not keywords as these are the only values for Boolean data type)

ii. **byte**
   1 signed byte (two's complement). Covers values from -128 to 127.

iii. **short**
   2 bytes, signed (two's complement), -32,768 to 32,767

iv. **int**
   4 bytes, signed (two's complement). -2,147,483,648 to 2,147,483,647. Like all numeric types int may be cast into other numeric types (byte, short, long, float, double). When *lossy* casts are done (e.g. int to byte) the conversion is done modulo the length of the smaller type.

v. **long**
   8 bytes signed (two's complement). Ranges from -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807.

vi. **float**
   4 bytes, IEEE 754. Covers a range from 1.40129846432481707e-45 to 3.40282346638528860e+38 (positive or negative).
   Like all numeric types floats may be cast into other numeric types (byte, short, long, int) the fractional part is truncated and the conversion is done modulo the length of the smaller type.

vii. **double**
   8 bytes IEEE 754. Covers a range from 4.94065645841246544e-324d to 1.79769313486231570e+308d (positive or negative).

viii. **char**
   2 bytes, unsigned, Unicode, 0 to 65,535
   Chars are not the same as bytes, ints, shorts or Strings.

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

3.3 Summary table of primitive data type

Data Type	Purpose	Contents	Default Value
boolean	Truth value	true or false	false
char	Character	Unicode characters	\u0000
byte	Signed integer	8 bit two's complement	(byte) 0
short	Signed integer	16 bit two's complement	(short) 0
int	Signed integer	32 bit two's complement	0
long	Signed integer	64 bit two's complement	0L
float	Real number	32 bit IEEE 754 floating point	0.0F
double	Real number	64 bit IEEE 754 floating point	0.0D
Object	Represents predefined any class		null
String	Represents String class		null

3.4 Reference Data Types

In Java a **reference data type** is a variable that can contain the reference or an address of dynamically created object. This type of data type is not predefined like **primitive** data type. The reference data types are **arrays**, **classes** and **interfaces** that are made and handle in a java program as per the need of the program.

array type //Points to an array instance
class type //Points to an object or a class instance
interface type //Points to an object which is implemented to the corresponding interface

3.4.1 Class Type

As we know that Java is an object-oriented programming language where an object is a variable, associated with methods that is described by a class. The name of a class is treated as a **type** in a java program, so that we can declare a variable of an object-type, and a method which can be called using that object- type variable.
Whenever a variable is created, a reference to an object is also created using the name of a class for its type i.e. that variable can contain either **null** or a **reference** to an object of that class. The object becomes an **instance** when the memory is allocated to that object using **new** keyword.

3.4.2 Array Type

An array is a special kind of **object** that contains values called **elements**. The java array enables the user to store the values of the same type in **contiguous** memory allocations. The elements in an array are identified by an **integer index** which initially starts from **0** and ends with **one less than number** of elements available in the array. All elements of an array must contain the same type of value i.e. if an array is a type of integer then all the elements must be of integer type. It is a **reference data type** because the class named as **Array** implicitly extends **java.lang.Object**. The syntax of declaring the array is shown as:

DataType [ ] variable = new DataType [ArraySize];
DataType [ ] variable = {item 1, item 2,...item n};

For example:

```
int [ ] a = new int [10];
String [ ] b = {"reference","data", "type"};
```

In the first statement, an array variable **"a"** is declared of integer data type that holds the memory spaces according to the size of int. The index of the array starts from **a[0]** and ends with **a[9]**.
In the second statement, the array **"b"** is declared of string data type that has the enough memory spaces to directly hold the three string values. Thus each value is assigned for each **index** position of the array.

3.4.3 Interface Type

Java provides another kind of **reference data type** or a mechanism to support **multiple inheritance** features called an **interface.** The name of an interface can be used to specify the type of a reference. A value is not allowed to be assign to a variable declared using an interface type until the **object** implements the specified **interface**.

4. Wrapper Classes

4.1 What is wrapper class?

Wrapper classes are part of Java's standard library **java.lang** and these classes helps in converting the primitive data types into an object. A wrapper class wraps a value of primitive type in an object.

4.2 What is the need to wrapper class?

Reasons for uses of wrapper classes around the primitives are given below:

1. There are several common methods – **ClassType, typeValue, to String** etc. those are standard throughout each object.
2. **typeValue** method is used in converting from character strings to numbers int, byte, short, long, float and double
3. The **ClassType** method is the constructor for the wrapper classes. It simply takes an argument of the class type it is wrapping.
4. A way to store primitives in an object.

4.3 Different wrapper classes along with their methods

Method	Use
1. The **Integer Class** is wrapper class for the primitive data type **int**. The Integer class wraps an int value in an object	
static int parseInt(String)	It converts a String to an int data type
static Integer valueOf(String)	It converts a String to an Integer object
static String toString(int)	It converts an int to a String object
2. The **Float Class** is wrapper class for the primitive data type **float**. The Float class wraps a float value in an object	
static float parseFloat(String)	It converts a String to an float data type
static Float valueOf(String)	It converts a String to an Float object
static String toString(float)	It converts a float to a String object
3. The **Double Class** is wrapper class for the primitive data type **double**. The Double class wraps a double value in an object	
static double parseDouble(String)	It converts a String to a double data type
static Double valueOf(String)	It converts a String to an Double object
static String toString(int)	It converts an int to a String object
4. **Similar methods are there for Byte, Short and Long Class**	
5. The **Character Class** is a wrapper class for the primitive data type **char**. The Character class provides several methods, some of the commonly used methods are given below	
static boolean isDigit(char)	It returns **true** if the character data is a number digit else returns **false**.
static boolean isLetter(char)	It returns **true** if the character data is an alphabet else returns **false**.
static boolean isLetterOrDigit(char)	It returns **true** if the character data is an alphabet or a number digit else returns **false**.
static boolean isLowerCase(char)	It returns true if the character data is in lowercase i.e from 'a' to 'z' else returns false.
static boolean isUpperCase(char)	It returns true if the character data is in uppercase i.e from 'A' to 'Z' else returns false.
static boolean isWhiteSpace(char)	It returns true if the character data is a whitespace else returns false.
static char toUpperCase(char)	It returns a uppercase equivalent of the same, if it is an alphabet else no change
static char toLowerCase(char)	It returns a lowercase equivalent of the same, if it is an alphabet else no change

5. Expressions and Type Expressions

5.1 What is an expression?

Valid combination of operators and operands (variables or constants) that yields an output is known as expression. E.g. SI = (P*R*T)/100.0; here P,R,T are the variables to hold input value and SI is the variable to store output value. We see a set of operators in the above expression, that helps us to compute the answers.

5.2 Types of expression

Depending upon the operators, operands, data types etc we have different categories of expressions.

- **Arithmetic expression** – This type of expression contains arithmetic operation as the main task i.e. the expression is formed from integer or floating point operands and arithmetic operators.
  A=2+3+5;
  - **Pure expression** – This type of expression contains all the operands of the same data type.
    - **Sum = 2+3+5**;
  - **Mixed expression** – This type of expression contains different types of operands.
    - **Avg = (2+3+4)/2**.0;
- **Conditional expression or test-expression** – This type of expression contains test conditions as the main task i.e. the expression is formed from integer or floating point operands and relational operators. A>B or 5>3 or (a+b+c)>=100 or (a-b)<0
- **String concatenation** – This type of expression contains arithmetic operator (+) and the operands of String type. The + operator performs string concatenation only if at least one of its operands is of String type, otherwise it performs addition with primitive types. "ABC" + "123". The result will be ABC123.
  If the statement is "Sahil" + " " + "Agarwal", the result will be "Sahil Agarwal"

5.3 Type Conversion

Conversion of one data type into another data type is known as type conversion. Type conversion is useful to prevent loss of data or to fit a data of larger type into a variable of smaller type. There are two types of conversion – Implicit and Explicit. E.g - Avg=(2+3+4)/2.0; Ans. 4.5 (fractional value)

5.3.1 Implicit Type conversion

It is a conversion of a smaller type of variable into a higher data type present in the expression. This type of conversion takes place automatically to prevent loss of data during the execution of an expression. Another name of implicit conversion is **coercion**.

Given below is an example of implicit conversion-

```
int a=5;
double b=6.5;
double res= a+b;        the result of the statement will be 11.5 (double)
```

5.3.2 Explicit Type conversion

It is a forcefully conversion of one data type to another data type made by the programmer in the program. Explicit conversion is applicable for both – from lower to higher type and higher to lower type. This type of conversion is useful in preventing loss of data or to fit a data of larger type into a variable of smaller type when there is a type miss-match in the left-hand side operand and right-hand side operand of the assignment operator. Explicit type conversion is also known as **type casting**.

General syntax of explicit conversion: **(data type) operand**
Given below some examples of explicit type conversion-

```
int a=5, b=3;
double c;
c = b/a;
```

In the above expression we can see that the result will be 0.0 since division takes place on integer data type which gives 0 and then the value will be stored as 0.0 in the variable c. This is loss of precision since the same division in floating point will be 0.6. To prevent such loss of data, the change in the above expression will be as follows-

```
c = (double) b/a;        //forcefully conversion of b(int) to double type.
```

c = 0.6 (As the answer)

```
double a =10.5;
int b = (int) a; => (int) 10.5 = 10
double c = a – b; => 10.5 – 10 = 0.5
```

5.3.3 Difficulties in type conversion

The following types of problems occur during the type conversion:

1. If a bigger floating-point type is converted to a smaller floating-point type (i.e. double to float), then in that case loss of precision occurs.
2. If a floating-point type is converted to integer type, then in that case loss of fractional part occurs.
3. If a bigger integer type is converted to smaller integer type (i.e. long to short etc), then in that case loss of precision occurs.
4. In all the above cases, we observed that the original value may be out of range for the target type in which case result is undefined.

SUMMARY

1. What is datatype?

   Ans: Data type represents the type of data being operated upon. For example, the value 35.93749449 is of type double. So, it defined as double d = 35.93749449;

2. What is composite datatype?

   Ans: Composite data types are built up from primitive dada type. A composite dada type not be directly composed of primitive dada type but ultimately, they can be decomposed into primitive dada types. For example, an array of arrays. Composite data types are also known as user defined types.

3. Explain why a class can be thought of as a user defined datatype?

   Ans: User designs the structure of a class based on the requirement of the program. A class encapsulates a set of primitive datatypes together to be used as single unit. Hence, a class can be thought of as a user defined datatype.

4. Can variable of composite type be member variable? Explain with example?

   Ans: Yes, a variable of composite type can be member variable. Below is an example to justify the point.

   ```
   public class Element {
   int id;
   String name;
   }
   public class ElementGroup {
   String name;
   Element [] elements;
   }
   ```

   The class Element defines a composite data type Element. Class ElementGroup has member variable'name' of type.
   Sting and an array 'elements' of type Elements.

5. What is the space occupied by variable of composite datatype?

   Ans: the space occupied by a composite data type is the sum of spaces occupied by the data types it is composed of.

6. Explain what happen when the following statement is executed

Student s1 = new Student ();
Ans: The statement results in creation of an instance s1 of the class Student and allocation of memory to the instance.

7. What is wrapper class? Give example?
   Ans: Java provides Wrapper classes for the primitive dada type. For example, class integer warps up the primitive data type int. the wrapper classess can be used to crate objects that would be similar to primitive data type.
8. Explain the purpose of new operator?
   Ans: the new operator allocates memory for an object at runtime and returns a reference of this memory.
9. List all the primitive data types in java?
   Ans: Java supports eight primitive data type namely char, byte, int short, long, float, double, and Boolean.
10. Explain the role of a class as a Java application?
    Ans: A class containing main method is referred to as an application class. main is the first method to get executed in program. All objects' creations and method invocations on them happen in this method.

MCQ with answers:

Which of the following statements is false about objects?
A. An instance of a class is an object
B. Object can access both static and instance data.
C. Object is the super class of all other classes
D. Object do not permit encapsulation

Right Answer: **D**

All the wrapper class (Integer, Boolean, Float, Short, Long, Double and Character) in java
A. are private
B. are serializable
C. are immutable
D. are finale

Right Answer: **D**

The code snippet if ("Welcome".trim() == "Welcome".trim()) System.out.println("Equals"); else System.out.println("Not Equal"); will _____
A. compile and display Equal
B. compile and display Not Equal
C. cause a compiler error
D. compile and display NULL

Right Answer: **C**

What is an aggregate?
A. An object with only primitive attributes.
B. An instance of a class which has only static methods.
C. An instance which has other objects.
D. None of the above.

Right Answer: **C**

Which methods can access to private attributes of a class?
A. Only static methods of the same class.
B. Only instance of the same class.
C. Only methods those defined in the same class.
D. Only classes available in the same package.

Right Answer: **C**

Which of the following is considered as a blue print that defines the variables and methods common to all of its objects of a specific kind?
A. Object
B. Class
C. Method
D. Real data types

Right Answer: **B**

What is the meaning of the return data type void?
A. An empty memory space is returned so that developers can utilize it.
B. Void returns no data type.
C. Void is not supported in Java.
D. None of the above.

Right Answer: **B**

A lower precision can be assigned to higher precision value in Java. For example. A byte type data can be assigned to int type.
A. True
B. False

Right Answer: **A**

What is the data type for the number 9.6352?
A. float.
B. double.
C. Float.
D. Double.

Right Answer: **B**

Which of the following statements are about the Java language is true?
A. Both procedural and OOP are supported in Java.
B. Java supports only procedural approach towards programming.
C. Java supports only OOP approach.
D. None of the above.

Right Answer: **A**

If result = 2+3*5, what is the value and type of result variable?
A. 17, byte.
B. 25, byte.

C. 17, int.
D. 25, int.

Right Answers: **C**

How many numeric data types are supported in Java?
A. 8
B. 4
C. 2
D. 6

Right Answers: **D**

Which of the statement is reserved word in java?
A. Run
B. Import
C. transient
D. Implement

Right Answers: **B**

Which of the following is not a return type?
A. boolean
B. void
C. public
D. String

Right Answer: **C**

UNSOLVED EXERCISE

1. What is data type? Name two categories present in Java Programming
2. What are composite data types? Give some examples.
3. Explain the use of the following operator:-
    a. new
    b. instanceof
    c. params
    d. type
4. Explain why class can be thought of as a user defined data type.
5. List all the primitive data types in Java along with their size and range.
6. What is wrapper class? Give example.
7. What is meant by public access to the members of a class?
8. What is meant by protected access to the members of a class?
9. What is meant by private access to the members of a class?
10. What is meant by default access to the members of a class?