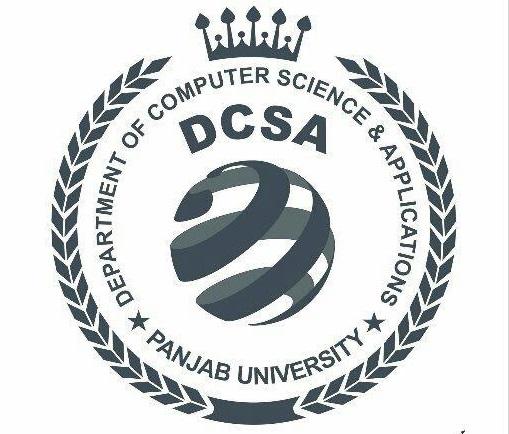
**DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS**

****

**PANJAB UNIVERSITY**

**CHANDIGARH**



JAVA BASIC ASSIGNMENT

**SUBMITTED TO: SUBMITTED BY:**

**Ms. Jasleen Kaur Bains**  **Abhay Singh**

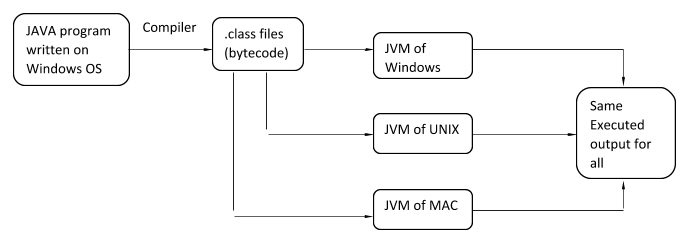
**MCA-II(Evening)**

**Roll No. 51**

**Theory Questions**

**Q1.**What makes JAVA a platform independent language?

**Ans:** The Java virtual machine make java a platform independent language as the java source code is compiled into intermediate code called byte code by the java compiler.

The byte code is platform independent, which can be easily interpreted by the JVM on any platform.

**Q2.**What all features of JAVA make it suitable for internet programming?

**Ans**: The features that makes java suitable for internet programming are:

* **Distributed:** Java provides extensive library of classed for communicating using TCP/IP protocol such as HTTP and FTP. This makes creating network communication much easier than c and c++.

It provides <applet> tag for html webpages for creating interactive webpages.

* **Secure:** As java is intended to be used in networked/distributed environment thus it implements several security mechanisms to protect from malicious code.

E.g. the absence of pointers makes it impossible for application to access the memory location without proper authorisation.

* **Multithreading:** Java can execute different parts of program multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
* **Robust:** The java language is reliable as it puts lot of emphasis on early checking of possible error’s which would be detected at runtime in other language.

Java forces the programmer to write code to deal with exception and it also supports runtime exception handling.

* **Architectural Neutral:** Java follows ‘Write once run anywhere’ approach.

Java programs are compiled into byte-code format which does not does not depend on any machine architectural but can easily translated into a specific machine instructions by JVM.

**Q3.**Why is JAVA called a compiled and interpreted language?

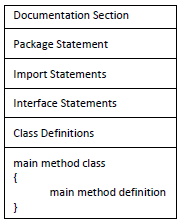
**Ans:** The Java programming language is both compiled and interpreted. The java compiler translates the java source code into the intermediate code called the byte code.

I.e. the .java file is converted into a .class file.

The class file can be interpreted by the java virtual machine running on any platform.

**Q4.** Discuss the structure of JAVA program.

**Ans:**



**1. Documentation Section:**  It is simply a message that exists only for the programmer and is ignored by the compiler. A good program should include comments that describe the purpose of the program, author name, date and time of program creation. This section is optional and comments may appear anywhere in the program.

Java programming language supports three types of comments.

* **Single line (or end-of line) comment**: It starts with a double slash symbol (//) and terminates at the end of the current line. The compiler ignores everything from // to the end of the line. For example:

***//*Calculate sum of two numbers**

* **Multiline Comment**: Java programmer can use C/C++ comment style that begins with delimiter /\* and ends with \*/*.*
* **Documentation comments**: This comment style is new in Java. Such comments begin with delimiter */\*\**and end with \**/.*

The main purpose of this type of comment is to automatically generate program documentation.

**2. Package Statement**: Java allows you to group classes in a collection known as package. It must appear as the first statement in the source code file before any class or interface declaration.

***E.g. package employee;***

This statement declares that all classes and interfaces defined in this source file are part of the employee package.

Only one package declaration can appear in the source file.

**3. Import Statement**: An import statement is used for referring classes that are declared in other packages. The import statement is written after a package statement but before any class definition. You can import a specific class or all the classes of the package.

For example: If you want to import Date class of *java.util* package using import statement then write

***import java.util.Scanner;***

Unlike package statement, you can specify more than one import statement in your program.

**4. Interface Section**: In the interface section, we specify the interfaces. An interface is similar to a class but contains only constants and method declarations. Interfaces cannot be instantiated. They can only be implemented by classes or extended by other interfaces. It is an optional section and is used when we wish to implement multiple inheritance feature in the program.

**interface** myImplements {  
    
}

**5. Class Section**: The Class section describes the information about user-defined classes present in the program. A class is a collection of fields (data variables) and methods that operate on the fields. Every program in Java consists of at least one class, the one that contains the main method.

The main () method which is from where the execution of program actually starts and follow the statements in the order specified.

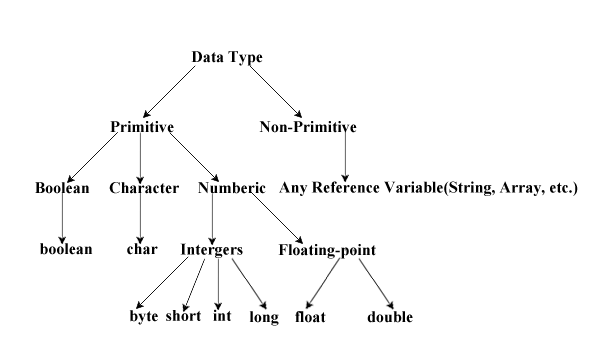
The main method can create objects, evaluate expressions, and invoke other methods and much more. On reaching the end of main, the program terminates and control passes back to the [operating system](http://ecomputernotes.com/fundamental/disk-operating-system/what-is-operating-system).

***The class section is mandatory.***

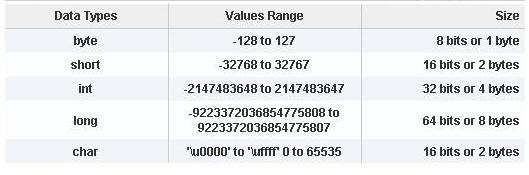
*class Test {  
  public static void main(String abc[]){  
   System.out.println("Program");  
   }  
}*

**Q5.**List the various data types used in JAVA.

**Ans :** Following are the basic data types in Java:



Range of the data types:



**Q6**.Discuss the various unary and binary operators used in JAVA.

**Ans:** The various types of operators availabe in Java which are given below:

* Unary Operator,
* Arithmetic Operator,
* Shift Operator,
* Relational Operator,
* Bitwise Operator,
* Logical Operator,
* Ternary Operator and
* Assignment Operator.

**1.Unary Operators:** Unary operators need only one operand. They are used to increment, decrement or negate a value.

* **– : Unary minus**, used for negating the values.
* **+ : Unary plus**, used for giving positive values. Only used when deliberately converting a negative value to positive.
* **++ : Increment operator**, used for incrementing the value by 1. There are two varieties of increment operator.
  + **Post- Increment :** Value is first used for computing the result and then incremented.
  + **Pre- Increment :**Value is incremented first and then result is computed.
* **— : Decrement operator**, used for decrementing the value by 1. There are two varieties of decrement operator.
  + **Post-decrement :**Value is first used for computing the result and then decremented.
  + **Pre-Decrement :**Value is decremented first and then result is computed.
* **! : Logical not operator**, used for inverting a boolean value.

**2. Arithmetic Operators:** They are used to perform simple arithmetic operations on primitive data types.

* **\* :**Multiplication
* **/ :**Division
* **% :**Modulo
* **+ :**Addition
* **– :**Subtraction

**3. Assignment Operator : ‘=’** Assignment operator is used to assign a value to any variable. It has a right to left associativity.

In many cases assignment operator can be combined with other operators to build a shorter version of statement called **Compound Statement**. For example, instead of a **=** a+5, we can write a **+=** 5.

* **+=**, for adding left operand with right operand and then assigning it to variable on the left.
* **-=**, for subtracting left operand with right operand and then assigning it to variable on the left.
* **\*=**, for multiplying left operand with right operand and then assigning it to variable on the left.
* **/=**, for dividing left operand with right operand and then assigning it to variable on the left.
* **%=**, for assigning modulo of left operand with right operand and then assigning it to variable on the left.

**4 .Relational Operators:** These operators are used to check for relations like equality, greater than, less than. They return Boolean result after the comparison and are extensively used in looping statements as well as conditional if else statements.

Some of the relational operator’s are-

* **==, Equal to :**returns true of left hand side is equal to right hand side.
* **!=, Not Equal to :**returns true of left hand side is not equal to right hand side.
* **<, less than :**returns true of left hand side is less than right hand side.
* **<=, less than or equal to :**returns true of left hand side is less than or equal to right hand side.
* **>, Greater than :**returns true of left hand side is greater than right hand side.
* **>=, Greater than or equal to:**returns true of left hand side is greater than or equal to right hand side.

**5. Logical Operators :** These operators are used to perform “logical AND” and “logical OR” operation, i.e. the function similar to AND gate and OR gate in digital electronics.   
Conditional operators are-

* + **&&, Logical AND :**returns true when both conditions are true.
  + **||, Logical OR :**returns true if at least one condition is true.

**6.Ternary operator :** Ternary operator is a shorthand version of if-else statement. It has three operands and hence the name ternary.

Syntax: condition ? if true: if flase

e.g (10>20)? System.out.println (10):System.out.println (20);

**7. Bitwise Operators:** These operators are used to perform manipulation of individual bits of a number. They can be used with any of the integer types. They are used when performing update and query operations of Binary indexed tree.

* **&, bitwise AND operator:**returns bit by bit AND of input values.
* **|, Bitwise OR operator:**returns bit by bit OR of input values.
* **^, Bitwise XOR operator:**returns bit by bit XOR of input values.
* **~, Bitwise Complement Operator:**This is a unary operator which returns the one’s compliment representation of the input value, i.e. with all bits inversed.

**8. Shift Operators:** These operators are used to shift the bits of a number left or right thereby multiplying or dividing the number by two respectively. They can be used when we have to multiply or divide a number by two.

* **<<, Left shift operator:**shifts the bits of the number to the left and fills 0 on voids left as a result. Similar effect as of multiplying the number with some power of two.
* **>>, Signed Right shift operator:**shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit depends on the sign of initial number. Similar effect as of dividing the number with some power of two.
* **>>>, Unsigned Right shift operator:**shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit is set to 0.

**9.** **Instanceof operator :** Instance of operator is used for type checking. It can be used to test if an object is an instance of a class, a subclass or an interface.  *E.g.obj1 instanceof Student*

**Q7.**Discuss the following control and iterative statements used in JAVA

* 1. If-elseif
  2. For-each loop in JAVA
  3. For statement

**Ans:**

**a) If-else :** It is a control statement that allow java program to execute the program statements by meeting certain test condition.

The **if**(test condition) block executes only if the text condition evaluates true else the else{} block is executed.

Syntax: ***if(test condition)***

***{ statements;}***

***else***

***{statements;}***

e.g ***if(10>20)***

***System.out.println(“10 is greater”);***

***else***

***System.out.println(“20 is greater”);***

**b)for-each loop:** It is enhanced for loop providing an alternative approach to traverse the array or collection in java.

It traverse each element of an array one by one and it can not traverse the element in reverse order as it does not work on index basis.

Syntax: ***for ( data type variable: array or collection){***

***Statements;***

***}***

e.g. ***int arr[]={1,2,4,5,6,7}***

***for (int i:arr){***

***System.out.println(i);***

***}***

**c) for loop:** It is a looping statement, and it contains initialisation statement, test condition and increment /decrement expression.

It is used in those situation where the programmer know in advance the number of time a set of statement to be executed.

Syntax: ***for (initialisation; condition; increment/decrement){  
 statements;***

***}***

***e.g. for(int i=0;i<11;i++){***

***System.out.println(i);//it print counting till 10***

***}***

**Practical Questions**

**Q8. /\*\* A JAVA program to implement arithmetic, bitwise and logical operators in JAVA**

**using switch case. Create a menu based program using do while loop,ask the user for**

**appropriate choice of operator,matching case block of switch case should be executed. \*/**

import java.util.Scanner;

class Operations{

public static void main(String abc[]){

Scanner sobj=new Scanner(System.in);

int opt;

do{

System.out.println("Please choose any option:");

System.out.println("1.Aritmetic Operations");

System.out.println("2.Bitwise Operations");

System.out.println("3.Logical Operations");

System.out.println("4.Close Program");

opt=sobj.nextInt();

switch(opt){

case 1:arithmeticOperartion();

break;

case 2:bitwiseOperation();

break;

case 3:logicalOperation();

break;

case 4: System.exit(0);

break;

default:System.out.println("Invalid choice!");

break;

}

}while(opt!=4);

}

static void arithmeticOperartion(){

int a,b;

Scanner sobj=new Scanner(System.in);

System.out.print("Enter first operand a:");

a=sobj.nextInt();

System.out.print("Enter second operand b:");

b=sobj.nextInt();

System.out.println("Aritmetic Operations");

System.out.println(a+"+"+b+" : "+(a+b));

System.out.println(a+"-"+b+" : "+(a-b));

System.out.println(a+"\*"+b+" : "+(a\*b));

System.out.println(a+"/"+b+" : "+(a/b));

System.out.println(a+"%"+b+" : "+(a%b));

}

static void bitwiseOperation(){

int a,b;

Scanner sobj=new Scanner(System.in);

System.out.print("Enter first operand a:");

a=sobj.nextInt();

System.out.print("Enter second operand b:");

b=sobj.nextInt();

System.out.println("Bitwise Operations");

System.out.println(a+"&"+b+" : "+(a&b));

System.out.println(a+"|"+b+" : "+(a|b));

System.out.println(a+"^"+b+" : "+(a^b));

System.out.println("~"+b+" : "+(~b));

System.out.println(a+"<<"+b+" : "+(a<<b));

System.out.println(a+">>"+b+" : "+(a>>b));

System.out.println(a+">>>"+b+" : "+(a>>>b));

}

static void logicalOperation(){

boolean a,b;

Scanner sobj=new Scanner(System.in);

System.out.print("Enter true or flase: ");

a=sobj.nextBoolean();

System.out.print("Enter true or false : ");

b=sobj.nextBoolean();

System.out.println("Logical Operations");

System.out.println(a+" && "+b+" : "+(a&&b));

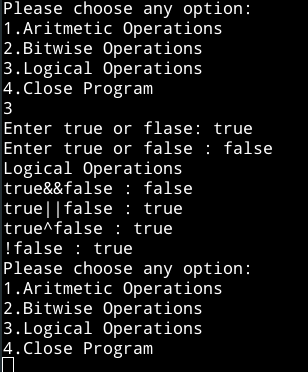
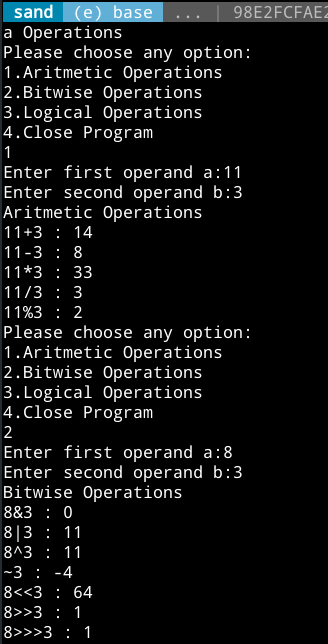
System.out.println(a+" || "+b+" : "+(a||b));

System.out.println(a+" ^ "+b+" : "+(a^b));

System.out.println(" !"+b+" : "+(!b));

}

}



**Q9. Create a JAVA program to check whether a number is odd or even number.**

*//Program to check a number is odd or even*

*import java.util.Scanner;*

*class OddEven{*

*public static void main(String abc[]){*

*Scanner sobj= new Scanner(System.in);*

*int num;*

*System.out.println("Please enter a number:");*

*num=sobj.nextInt();*

*if(num%2==0)*

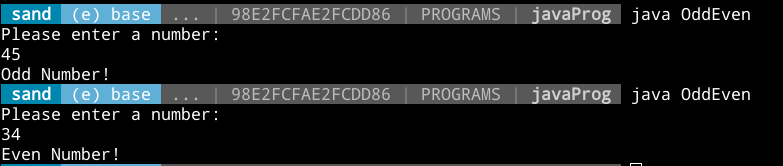
*System.out.println("Even Number!");*

*else*

*System.out.println("Odd Number!");*

*}*

*}*



***Q10. Create a JAVA program to create the following pattern***

\*

\* \*

\* \* \*

*import java.util.Scanner;*

*class Pattern{*

*public static void main(String [] abc){*

*Scanner sobj=new Scanner(System.in);*

*System.out.println("Enter the pattern height:");*

*int height=sobj.nextInt();*

*for(int i=0;i<height;i++){*

*for(int j=0;j<height-i;j++)*

*System.out.print(" ");*

*for(int k=0;k<2\*i-1;k++)*

*System.out.print("\*");*

*System.out.println("");*

*}*

*}*

*}*

