Name:	**Miandad Khan**
ID:	**14130**
Subject:	**OOP**
Date:	**30/9/2020**

Q1. How many variables are being supported by java justify your answer with the help java coded example for each variable?

Ans:-

A variable provides us with named storage that our programs can manipulate. Each variable in Java has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.
You must declare all variables before they can be used. Following is the basic form of a variable declaration

Here *data type* is one of Java's data types and *variable* is the name of the variable. To declare more than one variable of the specified type, you can use a comma-separated list.

Following are valid examples of variable declaration and initialization in Java −

Example

```
int a, b, c;         // Declares three ints, a, b, and c.
int a = 10, b = 10;  // Example of initialization
byte B = 22;         // initializes a byte type variable B.
double pi = 3.14159; // declares and assigns a value of PI.
char a = 'a';        // the char variable a iis initialized with value
'a'
```

This chapter will explain various variable types available in Java Language. There are three kinds of variables in Java −

- Local variables
- Instance variables
- Class/Static variables

Local Variables

- Local variables are declared in methods, constructors, or blocks.
- Local variables are created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor, or block.

- Access modifiers cannot be used for local variables.
- Local variables are visible only within the declared method, constructor, or block.
- Local variables are implemented at stack level internally.

There is no default value for local variables, so local variables should be declared and an initial value should be assigned before the first use.

Example with coded:-

```
Import java.util.Scanner;

public class LocalVariable{

//create class for LocalVariable

public static void main(string[] args) {

//auto generated method

int  a=10;                //this is local variable created

int  b=20;                //this is also local variable created

int  c=a+b;               //this is also local variable created

system.out.printIn("sum of a and b is: " +c);

//will print the value of "C" which is the sum of "a" and "b"

}

}
```

Output:

Sum of a and b: 30

Instance Variables

- Instance variables are declared in a class, but outside a method, constructor or any block.
- When a space is allocated for an object in the heap, a slot for each instance variable value is created.
- Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.

- Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.
- Instance variables can be declared in class level before or after use.
- Access modifiers can be given for instance variables.
- The instance variables are visible for all methods, constructors and block in the class. Normally, it is recommended to make these variables private (access level). However, visibility for subclasses can be given for these variables with the use of access modifiers.
- Instance variables have default values. For numbers, the default value is 0, for Booleans it is false, and for object references it is null. Values can be assigned during the declaration or within the constructor.

Instance variables can be accessed directly by calling the variable name inside the class. However, within static methods (when instance variables are given accessibility), they should be called using the fully qualified name. *ObjectReference.VariableName*

Example java coded expalin:-

```
Import java.util.Scanner;

Public class InstanceVariable {

//here we declare the instance variabe

Public void addition(){

Int     a=10;          //local variable a is declare

Int     b=20;          //local variable b is declare

c       =a+b;          //Here we store a and b in “c”

system.out.printIn(“Sum of  a and b is: c);

//Here will print “Sum of  a and b is:” after run the program

}

}
```

Output

Sum of a and b is: 30

Class/Static Variables

- Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
- There would only be one copy of each class variable per class, regardless of how many objects are created from it.
- Static variables are rarely used other than being declared as constants. Constants are variables that are declared as public/private, final, and static. Constant variables never change from their initial value.
- Static variables are stored in the static memory. It is rare to use static variables other than declared final and used as either public or private constants.
- Static variables are created when the program starts and destroyed when the program stops.
- Visibility is similar to instance variables. However, most static variables are declared public since they must be available for users of the class.
- Default values are same as instance variables. For numbers, the default value is 0; for Booleans, it is false; and for object references, it is null. Values can be assigned during the declaration or within the constructor. Additionally, values can be assigned in special static initializer blocks.
- Static variables can be accessed by calling with the class name *ClassName.VariableName*.

When declaring class variables as public static final, then variable names (constants) are all in upper case. If the static variables are not public and final, the naming syntax is the same as instance and local variables.

Example java coded expalin:-

```
Import java.util.Scanner;

Public class staticVariable{

Static String sum =”sum: “              //string which hold certain value that is sum

Static String diff =”Difference: “      //string which hold certain value that is diff

Public static void main(String[] args){

Int     a=10;           //here is int a which  value is 10

Int     b=20;           //here is int b which  value is 20

Int     c=a+b;          //here in c we sum the value of a and b
```

```
Int       d=a-b;            //here in d we diffe the value of a and b


Syste.out.printIn(sum+c);

//here will sum both the values

Syste.out.printIn(diff+d);

//here will diff both the values


}

}
```

Output:

30

10

Q2. Why "If" is used in java justify your answer with the help java coded example and explain in detail?

Ans:-

If Statement:-

If statement consists a condition, followed by statement or a set of statements as shown below:

Example:-

```
Import java.util.Scanner;
public class IfStatementExample {
        /* class States created */

public static voidmain(String args[]){
        /* 1000 auto generated method stub*/

 int num=70;
        /* Here input the Number */

 if( num <100>{
```

```
            /* This println statement will only execute,
             * if the above condition is true */

System.out.println("number is less than 100";
          /* this will produce in out */
    }
  }
}
```

Output:

Number is less than 100

Q3. Why "if else if" is used in java justify your answer with the help java coded example and explain in detail?

Ans :- IF ELSE statement:-

An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is false.

Example:-
Program for age restricted.

```
Import java.util.Scanner;
public class conduct {
//create a class for conduct

pubic static void main(String[] args )  {
//auto-generated method stub

Int       a;   //int take at as "a"

Scanner m=new Scanner(System.in);
//here I take the variable of m

system .out.println("enter your numb: ")
// here the user will enter their age after run the code
a=m.nextIn();
//here m which is variable will hold the int that is "a"

if(a<=20>
//here I give the condition for age to enter
system.out.println("You cannot enter: ");
//this will print if a is smaller or equal to 20
}
else(a>20)
```

```
//if a is greater than 20 then the this part will executed
{
system.out.printIn("You can enter: ");
}
}
```

Output:-
enter your numb:

Q4. What are loops, why they are used in java and how many types of loops are being supported by java explain in detail?

Ans:- LOOP:-

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A **loop** statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages −

Java programming language provides the following types of loop to handle looping requirements

TYPES OF LOOPS USED IN JAVA:-

There are three main loops used in JAVA, which are given as below

while loop

Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.

Example:-

```
Import java.util.Scanner;
class WhileLoopExample {
    public static void main(String args[]){
         int i=10;
//Here we put the value of I which is 10
         while(i>1){     //This is condition
```

```
                System.out.println(i);
                i--;         //here from the value it will decrements
                form the given value by minusing one
            }
        }
}
```

Output:

```
10
9
8
7
6
5
4
3
2
```

for loop

Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.

Example:-

```
// Program to print a text 5 times

class Main {
  public static void main(String[] args) {
    int n = 5;
    // for loop
    for (int i = 1; i <= n; ++i)
//Here we say to executed the for loop n time which is “5”
 {
      System.out.println("Java is fun");
    }
  }
}
```

Output

```
Java is fun
Java is fun
```

```
Java is fun
Java is fun
Java is fun
```

do...while loop

Like a while statement, except that it tests the condition at the end of the loop body.

```
Example:-
 class DoWhileLoopExample {
    public static void main(String args[]){
         int i=10;
                  //here initaialized the int i=10
         do{
              System.out.println(i);
              i--;
                  //Here loop decrement from the given value
         }while(i>1);
                  //here we give the minimum value to execute for
    }
}
```

Output:

```
10
9
8
7
6
5
4
3
2
```

why they are used in java:-

Imagine a program which is required to output a particular value of a variable 800 times. Now we all know that the code for writing output is System.out.println("Text");

But in order to print this 800 times we will need to write the same line 800 times in the code. That would take up a lot of effort which is particularly just copy pasting the same sentence 800 times. Let us say that you have managed to copy paste the entire thing easily.

Now if there is a different program which requires you to print the first 800 natural numbers. The copy paste method would not work because you still would have to go to all these lines and fit a number. That's where loops come in to play. Loops make it very easy to group all the code that's needed to be repetitively processed and throw it under scope. The loop does the remaining job

Q5. Write 3's table in decremented form in java which takes input from user write java coded program and explain in detail?

Ans:- 3's table in decremented form:-

Pogram:-

```
import java.util.Scanner;

public class Exercise7 {

 public static void main(String[] args) {
  Scanner in = new Scanner(System.in);

  System.out.print("Input a number: ");
//here input the number 3
  int num1 = in.nextInt();

  for (int i=10; i<10; i--){
//here I use the forloop to in decrement form
   System.out.println(num1 + " x " + (i-1) + " = " +
    (num1 * (i-1)));
//after giving the input the table will executed and will get the answer
  }
 }
}
```

Output

```
3 *10 =30
3 *9=27
3 *8 =24
3 *7 =21
3 *6 =18
3 *5 =15
```

3 *4 =12
3 *3 =9
3 *2 =6
3 *1 =3