Type: MCQ

Q1. what is the range of data type byte in Java? (0.5)

1. ** -128 to 127
2. -32768 to 32767
3. -2147483648 to 2147483647
4. None of these.

Q2. Which of the following class is used to convert date from one format to the other (0.5)

1. Date
2. SimpleDate
3. **SimpleDateFormat
4. DateConverter

Q3. Given two integer variables a and b. How do you test whether exactly one of them is zero. Identify the correct statement. (0.5)

1. (a == 0 && b != 0) &&(b == 0 && a != 0)
2. **(a == 0 && b != 0) || (b == 0 && a != 0)
3. (a == 0 || b != 0) || (b == 0 || a != 0)
4. (a == 0 || b != 0) && (b == 0 && a != 0)

Q4. Select the valid statement. (0.5)

1. **double[] db = new double[5]
2. double[] db = new double()
3. double[] db = new double []
4. double[] db = new double (5)

Q5. What will be stored in the variable x after executing the following statement:
int a = -1; int x = a++; (0.5)

1. 1
2. 0
3. **-1
4. -2

Q6. What will be the output of the following Java code?

```
public class Demo{
public static void main(String[] args)
   {
      byte i = 97;

     System.out.println((char)i);
   }
}
```

(0.5)

1. ** a
2. 1100001
3. A
4. ‘97’

Q7. Identify false statement about constructor. (0.5)

1. Constructor is a special type of method
2. ** Constructor can return a value
3. Constructor is used to initialize an object
4. Constructor can be overloaded

Q8. What is the minimum number of argument/s that can be passed to “public static void main(String[] args)”? (0.5)

1. 2
2. 1
3. **0
4. More than 2

Q9. How many superclasses can be inherited by a subclass? (0.5)

1. None of these
2. **Only one
3. two
4. Any number

Q10. What value is returned by compareTo() method in case the invoking string happens to be greater than the compared string? (0.5)

1. **a value that is greater than zero
2. a value that is less than zero
3. Zero
4. None of these

Type: DES

Q11. Create a class called Employee (instance fields: empID, name, age, Salary) with necessary constructor and methods corresponding to the Employee instances used in the EmployeeClassDemo

as shown in the Fig 1:

```
public class EmployeeClassDemo {
   public static void main(String[] args) {
      Employee[] staff = {  new Employee( 1, "Anil", 25, 50000 ),
                            new Employee( 2, "John", 35, 60000 ),
                            new Employee( 3, "Vinod", 38, 40000 )
                         };
      for( int i = 0 ; i < staff.length; i++ )
         staff[i].raiseSalary( 5 ); // Raise everyone's salary by 5%

      for( int i = 0 ; i < staff.length; i++ )
         System.out.println( staff[i] );
   }
}
```

OUTPUT:

```
Emp id: 1
Name: Anil
Age : 25
salary : 52500.0

Emp id: 2
Name: John
Age : 35
salary : 63000.0

Emp id: 3
Name: Vinod
Age : 38
salary : 42000.0
```

Fig 1: output screen for 11th question

(2)

Solution:

```
class Employee
{
   private int ID;
   private String name;
   private int age;
   private double salary;
   Employee( int id , String n, int a , double s )
   {
     ID = id; name = n;  age = a;  salary = s;
   }
   void raiseSalary( double byPercent )
   {
      double raise = salary * byPercent / 100;
      salary += raise;
   }
   public String toString()
   {
       return  "\n Emp id: "+ID+"\n Name: " + name  + "\n Age : "+ age + "\n salary : " + salary;
   }
}
```

Class with constructor → 0.5M

Raisesalary() → 0.5M

Overridden toString() → 1M

Q12. What will be content of arr1 and arr2 after executing the following statements. Justify your answer.

int arr1[]={ 11 , 22 , 33 } , arr2[]={ 99 , 88 , 77 };

arr1 = arr2; arr2[1] = 7; (2)

Solution:

arr1 : { 99 , 7 , 77 } → 0.5 mark

arr2 : { 99 , 7 , 77 } → 0.5 mark

Justification: In java , array is of reference type. So, both arr1 and arr2 point to the same reference. → 1 mark

Q13. Create a class "Literature" with two attributes as "title" and "author" and a method as "print()" without any definition. Now extend two classes from Literature class as "Book" and "Poem". Let "Book" class hold attributes as "publisher" and "genre", with a method "print()". Let "Poem" class hold attribute "style" with a method "print()". Implement the above scenario using dynamic *method dispatch*. Define parameterized constructors in all the 3 classes. Justify how dynamic method dispatch is achieved. (3)

Solution:

```
abstract class Literature
{
        String title , author;
        Literature( String title , String author )
        {
                this.title = title;
                this.author = author;
        }
        abstract void print();
        Literature()
        {}
}

class Book extends Literature
{
        String publisher , genre;
        Book(String title,String author,String publisher,String genre)
        {
                super( title , author);
                this.publisher = publisher;
                this.genre = genre;
        }
        Book()
        { }
        void print()
        {
          System.out.println("Title: "+title+"\nAuthor: "+author );
          System.out.println("publisher: "+publisher+"\ngenre: "+genre );
        }
}
```

```
class Poem extends Literature
{
        String style;
        Poem(String title,String author,String style)
        {
                super( title , author);
                this.style = style;
        }
        Poem()
        { }
        void print()
        {
                 System.out.println("Title: "+title+"\nAuthor: "+author );
                 System.out.println("style: "+style);
        }
}
public class LiteratureDemo
{
        public static void main(String args[])
        {
                Literature L = new Book("XYZ","John","Pearson","novel");
                System.out.println("Book Details: ");
                L.print();
                L = new Poem("The Road Not Taken","Robert Frost", "rhyme");
                System.out.println("\nPoem Details: ");
                L.print();
        }
}
```

Scheme:

Abstract class *Literature* with abstract method *print( )* → 1 mark
Book class with parameterized constructor → 0.5 mark
Poem class with parameterized constructor → 0.5 mark
Justification for dynamic method dispatch → 1 mark.

Q14. Given string objects, s1 be " Welcome " and s2 be " welcome ". Write the statements for the following operations:

i) Check whether s1 is equal to s2, ignoring case, and assign the result to a Boolean variable isEqual.

ii) Create a substring of s1 from index 1 to index 4.

iii) Split "Welcome to Java" into an array tokens delimited by a space . (3)

Scheme:

i) `boolean isEqual = s1.equalsIgnoreCase(s2);` → 1 mark

ii) `String sub = s1.substring(1,5);` → 1 mark

OR

```
char sub2[] = new char[4];
s1.getChars(1,5,sub2,0);
```

iii) `String tokens[] = "Welcome to Java".split(" ");` → 1 mark

OR

```
String tokens[] = "Welcome to Java".split("\s+");
```