End - Sem Examination

Name :- Tanishq Chauhan

Roll No :- 21C3184

Subject :- Object Oriented Programming (CER3C2)

Semester :- 3rd Semester

Branch :- Computer Science & Engineering (CS-B)

Chauhan

Tanishq Chauhan
21C3184
CS-B

Chauhan

Date
Page 01

Object Oriented Programming (CER3C2)

* Answer of Q. No 1. (a)

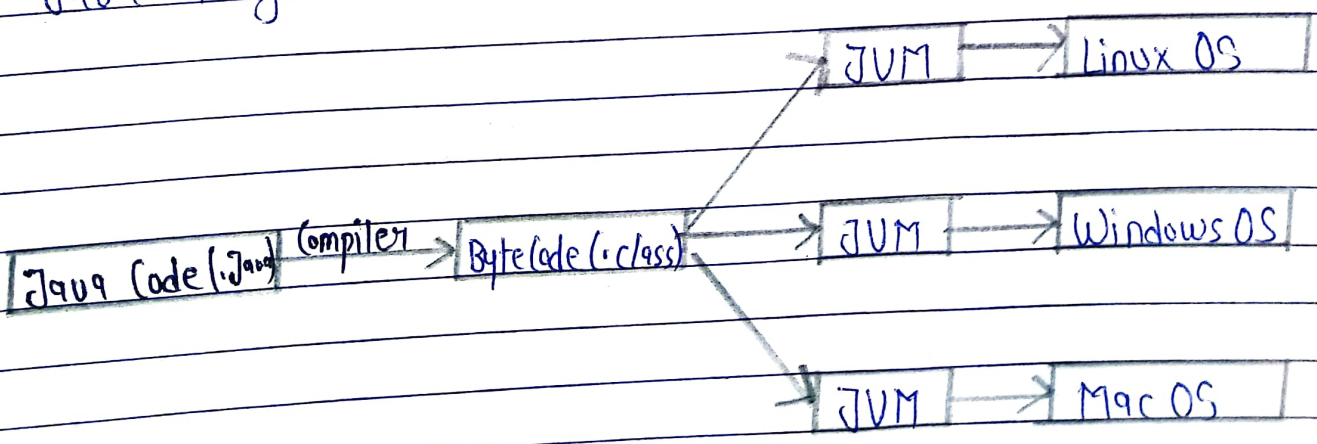
1. Java is platform neutral language

Java is Platform Neutral because the same Java code will run on multiple platforms (Operating System) without any modification, provided that the code does not intentionally put any specific demands on the system, holding true to the slogan, "Write Once, Run Anywhere".

Java's platform independence consists mostly of its Java Virtual Machine (JVM), which is a well specified and mature virtual machine.

On compilation Java program is compiled into Bytecode. This bytecode is platform independent and can be run on any machine, plus this bytecode format also provide security.

Any machine with Java Runtime Environment can run Java Programs.



Tanishq Chauhan
21C3184
CS-B

Chauhan
Date
Page 02

Answer of Q. No. 1 (c)

```
import java.util.Scanner;
public class ObjCount
{
    protected static int count = 0;

    public ObjCount()
    {
        count++;

        if (count<5)
                System.out.println("Too Less...");
        else if (count==5)
                System.out.println("Will Do...");
        else if (count>5)
                System.out.println("Exceeding The
                                    Limits...");
    }

    public static int getCount()
    {
        return count;
    }
```

Tanishq Chauhan
21C3184
CS-B


```
public static void main (String args[])
{
    ObjCount o1 = new ObjCount();
    ObjCount o2 = new ObjCount();
    ObjCount o3 = new ObjCount();
    ObjCount o4 = new ObjCount();
    ObjCount o5 = new ObjCount();
    ObjCount o6 = new ObjCount();

    int objectsCreated = getCount();

    System.out.println("Number of objects Created
                        are:" + objectsCreated);
    }
}
```

Output

```
Too Less...
Too Less...
Too Less...
Too Less...
Will Do...
Exceeding The limits...

Number of objects Created are: 6
```

Tanishq Chauhan
21C3184
CS-B

Chauhan

Page No. 04

Answer of Q. No 1. (9)

2. A constructor will have a access control of type default when no access modifier is defined explicitly. So this constructor will have a Package level Access. Classes which are defined within that package as that of the class with this default constructor will be able to access it and also the classes that extend this class containing the default constructor will be able to access it via inheritance.

If the constructor is made private, then only the code within that class can access this.

Example :-

```
public class Test
{
private static Test uniqueInstance = new Test();

private Test() {}

public static Test getInstance()
{
return uniqueInstance;
}
}
```

Constructor can't be static as it is called where object is created
Constructor can have any of the access modifier: public, protected, private or none (package or friendly).

Tanishq Chauhan
21C3184
CS-B

Chauhan

Date
Page 05

Answer of Q. No 1. (a)

3. Automatic type conversion in java takes place when two types are compatible and size of destination is larger than the source type.

For example :- In java, the numeric data types are compatible with each other but no automatic conversion is supported from numeric type to char or boolean. Also char and boolean are not compatible with each other.

Byte → Short → Int → Long → Float → Double

etc Example Code :-

```
class Conversion
{
    public static void main (String [] args)
    {
        int i = 100;
        long l = i;
        float f = l;

        System.out.println ("Int value:" + i);
        System.out.println ("Long value:" + l);
        System.out.println ("Float value:" + f);
    }
}
```

Tanishq Chauhan
21C3184
CS-B

Chauhan
Page 06

Output of the following code will be :-

Int value : 100
Long Value : 100
Float Value : 100.

Like above example we can convert the data types.

Tanishq Chauhan
21C3184
CS-B

Chauhan
Date
Page 07

Answer of Q No 2 (a)

Yes, we can substitute outer classes whereever we need to have inner classes but Inner classes have advantage in certain cases and hence preferred -

Ease :- Why ~~do~~ to implement a class outside if its objects are only intended to be part of an outer object. Its easy to define the class within another class if the use is only local.

Protection :- Making a call an outer exposes a threat of it being used by any of the class. Why should it be made an outer class if its objects should only occur as part of other objects.

For Example :- You may like to have an class address whose objects should have a reference to city and by design thats the only use of city you have in your application. Making Address and City as outer class exposes City to any of the class. Making it an inner class of Address will make sure that its accessed using object of Address.

Tanishq Chauhan
21C3184
CS-B

Difference between following code lines :-

1. new OuterClass().new InnerClass();
2. new OuterClass().InnerClass();

In first case we are trying to initialize Inner class object using the instance of Outer class whereas in second case we are trying to initialize the Inner class object directly using the Outer class name.

In second case, Inner class is "static inner class" as we cannot access "non static inner class" using classname alone.

In first case, the inner class could be either "static inner class" or "non static inner class".

So, following are the difference between the following 1 and 2 code lines

Tanishq Chauhan
21C3184
CS-B

Chauhan
Date
Page 09

Answer of Q. No 3 (b)

Runtime polymorphism in Java is also known as Dynamic polymorphism. In the process, the call to an override method is resolved dynamically at runtime rather than compile time.

Runtime polymorphism is achieved through.
Method Overriding

Mandatory Rules :-

- Same method name
- Same arguments (type, number, sequence)
- Same return type.
- It must not have a more restrictive access modifier than parent
  (if parent is protected then child → private not allowed)

- It must not throw new or broader ~~the~~ checked exception.

Only inheritated classes can be override.

By default, the type of object decodes which version of override method will be called but we can use reference variable of super class to call desired version of method. The type of object refered to by reference decides which version of overrider method will be called.

Tanishq Chauhan
2103184
CS-B

Chauhan

Lab Test 10.

Example :- we are creating two classes Bike and Splendor. Splendor class extends Bike class and overrides its run() method. We are calling the run method by the reference variable of Parent class. Since its refers to the subclass object and subclass method overrides the Parent class method, the subclass method is invocked at runtime.

Since method invocation is determined by the JVM not compiler, it is known as runtime polymorphism.

Code :-

```
class Bike    // Super Class
{
   void run()
      {
         System.out.println("Running");
      }
}

class Splendor extends Bike    // Sub Class
{
   void run()    // run method override
   {
      System.out.println("Running Safely with 60 Km");
   }
}
```

Tanishq Chauhan
21C3184
CS-B


```
class TestExample
{
    public static void main (String args[])
    {
        Bike b = new Bike();;
        Splendor s = new Splendor();

        Bike bi;    // reference variable

        bi = b;
        bi.run();  // call super class run method

        bi = s;
        bi.run;  // call - run method of sub class
    }
}
```

Output

Running

Running Safely with 60Km

Tanishq Chauhan
21C3184
CS-B

Answer of Q. No 2 (b)

```
import java.util.*;
import java.io.*;

public class AddElementsToVector
{
   public static void main (String[] arg)
   {

      Vector v1 = new Vector();

   // Adding element using add() method

      v1.add (10);
      v1.add (20);
      v1.add ("Hello");
      v1.add ("India");
      v1.add (2.20);
      v1.add (5.30);
      v1.add (8.90);

      System.out.println ("Before Removing Element at
                           3rd Position: " + v1);

      System.out.println ("Vector:" + v1);

      System.out.println ("After Removing Element:");
```

Tanishq Chauhan
21C3184
CS-B


```
v1.remove(2);
System.out.println("Vector: " +v1);

v1.t

v1.insertElementAt("New Element", 4);

System.out.println("Updated Vector : \n" +
                                    v1);
}
}
```

Output

Before Removing Element at 3rd Position :
Vector : [10, 20, Hello, India, 2.20, 5.30, 8.90]

After Removing Element :
Vector : [10, 20, India, 2.20, 5.30, 8.90]

Updated Vector :
Vecto
[10, 20, India, 1.2, New Element, 2.2, 3.3]

This will be the expected output of the following code.

Tanishq Chauhan
21C3184
CS-B

Chauhan
Date
Page 14

Answer of Q. No 3 (a)

1. A constructor is a special method of class in OOP that initializes a wholy created object of that type.

We can create constructor in abstract class because:

- Abstract classes have instance variables & abstract methods that need to be initialized which can be done using a constructor.
- Abstract classes have a default constructor needed for initializing of new objects.
- Abstract class has sub-classes, function of constructor of abstract class can be used by them.

We can't create constructor in interface because:

- Data member of interface are by default public, static and final and they used to be initialised during declaration which resolves the used to constructor.
- Interface methods are public abstract which means method implementation has to be provided by implementing class used of constructor.
- There are no non-static data members.
- Methods are not defined and thus no constructor is needed to make objects for calling there methods.

Tanishq Chauhan
21C3184
CS-B

2. For a class to invoked directly from the command prompt, it should have static members.

Inside inner class, static members can not be declared and this it is not possible to declare main() method & inner class cant be invoked directly from command prompt.

Example :-

```
class Outer
{
    Class Inner
    {
        public static void main (String[] args)
        {
            System.out.print ("Example");
        }
    }
}
```

Output :- Inner class cannot have static declaration.

Inside static nested class we can declare static members including main() method and thus it can invoked directly from Command Prompt.

Tanishq Chauhan
21C3184
CS-B

Example :-

```
class Example
{
    static class Nested
    {
        public static void main (String [] args)
        {
            System.out.println ("Nested Example");
        }
    }

    public static void main (String [] args)
    {
        System.out.println ("Outer Example");
    }
}
```

Output :-

```
Java Example
Outer Example
Java Example $ Nested
Nested Example
```

Tanishq Chauhan
21C3184
CS-B

Chauhan

Date
Page 17

Answer of Q. No 5.(a)

1. Non-Access Modifiers

Non-Access type of modifiers are used to control a variety of things, such as inheritance capabilities, whether all objects of our class share the same member value or have their own values of those members, whether a method can be overridden in a subclass etc.

Java provides a number of non-access modifiers to achieve many other functionalities.

(i) Static

- The static modifier for creating class methods and variables.
- The member belongs to the class, not to object of that class.

(ii) Final

- The final modifier for finalizing the implementations of classes, methods, and variables.
- A Variable values can't be changed once assigned, methods can't be overriden, classes can't be inherited.

Tanishq Chauhan
21C3184
CS-B

(iii) Abstract

- The abstract modifier for creating abstract classes and methods.
- If applied to a method - has to be implemented in a subclass, if applied to a class contains abstract method.

(iv) Synchronized

- The synchronized and volatile modifiers, which are used for threads.
- Controls thread access to a block/method.

Tanishq Chauhan.
21C3184
CS-B

2. Garbage Collection

- Java garbage collection is the process by which Java programs perform automatic memory management.
- Java programs compile to bytecode that can be run on a Java Virtual Machine, or JVM for short.
- When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program.
- Eventually some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory.
- To do so, we were using free() function in C language and delete in C++. But, in java it is performed automatically. So Java provides better memory management.

3. Demon Thread

Daemon thread in Java is a service provider thread that provides services to the user thread. Its life depend on the mercy of user threads i.e. when all the user threads dies, JVM terminates thread automatically.

Tanishq Chauhan
21C3184
CS-B

Chauhan

Date
Page 20

- There are many Java daemon threads running automatically e.g. gc, finalizer etc.
- You can see all the detail by typing the jconsole in the command prompt.
- You can use the setDaemon(boolean) method to change the Thread daemon properties before the thread starts.

Tanishq Chauhan
21C3184
CS-B

Chauhan 21

Answer of Q. No. 5 (c)

```
import java.~~util~~io.*;

public class Main
{
  public static void main (String args[])
  {
    int count = 0;
    File myFile = newfile ("test.txt");

    try {
        Scanner sc = new Scanner (myFile);

        while (sc.haNext())
        {
          String word = s(.next();
          if (word.equals("at"))
          {
            count ++;
          }
        }
        catch (Exception)
        {
          System.out.println (e); }
        }
        System.out.println ("Word at count is : "
                            + count);
    }
}
```