Name: Shadab Iqbal

ID : 1910107 2

Sec : 01

## Ans. to Q No - 1

## A

We need to overwrite the default constructor when we need to pass some values in the argument while initializing an object.

Example:

```
class Test {

    public int x;

    public Test() {
    }
    public Test(int a) {
        this.x = a;
    }
}
```

So, we can see from this example that if we want to initialize the value of x during an object creation, we will be needing the overriden constructor to do so.

# B

- Exception is an event that disrupts the normal flow of the program's instruction and occurs during the execution of a program. Exceptions are used so that the program does not terminate abruptly, but is pre-defined by the user the actions needed to be taken given that an unfavorable situation occurs.

Compile-time errors are generally syntax or Semantics error, gets detected by compiler before execution and can be fixed at the time of developing a code. Runtime errors occur during execution, and needs to be fixed after code gets executed and errors get identified.

And exceptions are used in Java custom classes to handle these runtime errors.

We might need multiple catch blocks in exceptions because there are many kinds of exceptions i.e ArithmeticExceptions, IOException and so on. So, using multiple catch blocks, we are pre-defining that those particular kinds of exceptions might occur while running the program and are needed to be caught.

# Ans. to Q.No - 2

## A

The main purpose of synchronization is the sharing of resources without interference. There are many uses and advantages of using it.

Scenario: Suppose we are designing a database system where at a time, only one user can see and update the database. If we use Synchronization here, we can make the current user to have a "lock" on the database so that no other user can update the database at that particular time.

## B

If a class is written as a "default" modifier, only other classes from the same package will have the privileges to access the class. Here, it is to be noted that any other classes (including subclasses), which are outside the package don't have the privilege to access that default class.

## C

If the only constructor is "private", we can not make any object of that class from other classes directly. If we want to create an object, there has to be a static method in the class which

will return an object of that class. But here it is to be noted that, we can create object of that class directly, only from inside of that class.

D

Daemon threads is a service provider thread that provides service to the user thread. It is a low priority thread and it's life depends on user threads. The sole purpose of daemon threads is to provide services to user threads for background supporting tasks. So, if there is no user threads, JVM terminates the daemon thread. Garbage Collector in Java is an example of daemon thread.

E

This will not work as it is, because 'C' is the parent class and 'B' is the child class.

We can not ~~~~ create an object of a parent ~~child~~ class referenced to a child class. So, what we can do here is :-

```
C obj = new B();
Obj.method();
```

Now the code will run, given that there is a method named "method" in the class C.

F

Method overloading: is a feature that allows a class in Java to have multiple methods with same name but different parameters.

We can distinguish methods having same name with these features below:

i) Parameter types.

ii) Parameter order.

iii) Number of parameters.

Method overriding: When a method in the subclass has the same name, parameters and same return type as a method in the parent class, then this is termed as method overriding.

## Ans. to Q No-3

```
public class ShapeFactory {

    public static Shape getInstance(String s, double d){

        if (s.equals("c")) return new Circle (d);

        if ( s.equals("s")) return new Square (d);

        return null;
    }
```

```
    public Static Shape getInstance (String s,
                                     double d1,
                                     double d2) {

        if (s.equals("r")) return new Rectangle (d1,
                                                 d2);

        return null;

    }

}
```

```
import static java.lang.Math.PI;

public class Circle implements Shape{
    private double r;
    public Circle (double d){
        this.r = d;
    }
    public double getArea() {
        return PI * r * r;
    }
```

```
    public double getPerimeter() {
        return 2 * PI * r;
    }

    public String toString () {
        return "circle";
    }
}
```

```
public class Square implements Shape {

    private double x;

    public Square(double d) {
        this.x = d;
    }

    public double getArea() {
        return x * x;
    }

    public double getPerimeter() {
        return 4 * x;
    }

    public String toString() {
        return "Square";
    }

}
```

```
public class Rectangle implements Shape {

    private double x;
    private double y;

    public Rectangle (double d1, double d2) {
        this.x = d1;
        this.y = d2;
    }

    public double getArea() {
        return x * y;
    }

    public double getPerimeter() {
        return 2 * (x+y);
    }

    public String toString () {
        return "Rectangle";
    }
}
```