Subhi Bhuiyan ID: 17201116 Time 4:30PM - 6PM

Answer to Question 1

A. The java toString() method automatically runs when an object is printed.

For example,

```
public class test {
    public static void main (String [] args) {
        Employee x = new Employee ();
        System.out.println (x);
    }
}

public class Employee {

}
```

8. An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

Exception provides the means to separate the details of what to do when something out of the ordinary happens from the main logic of a program.

The error indicates a problem that mainly occurs due to the lack of system resources and the application should not catch these types of problems. Errors mostly occur at runtime and belong to unchecked type.
Example: java.lang.StackOverflowError,
java.lang.OutOfMemoryError

Exceptions are the problems which can occur at runtime and compile time. Classified as checked and unchecked.
Example: ArrayIndexOutOfBoundException, NullPointerException.
Exceptions are mainly caused by the application itself.

The finally block always executes when the try block exits. This ensures that the finally block is executed even if an unexpected exception occurs.

Answers to Question 2

D. Daemon thread is a low priority thread that runs in the background to perform tasks such as garbage collection. Its life depends on users threads, i.e. when all users threads dies, JVM terminates this thread automatically.
The sole purpose of the daemon thread is that it provides services to user thread for background supporting task.
~~JVM termin~~

C. The use of private constructor is to serve singleton classes. By using private constructor, we can ensure that no more than one object can be created at a time. By providing a private constructor, we prevent class instances from being created in any place other than this very class.

F. Method Overloading is a feature that allows a class to have more than one method having the same name, but with different argument lists.
It is an example of Static Polymorphism.

Declaring a method in subclass which is already present in parent class is known as Method Overriding.
Overriding is done so that a child class can give its own implementation to a method which is already provided by the parent class. It is an example of Dynamic Polymorphism.

A. The need for synchronization originates when processes need to execute concurrently. The main purpose of synchronization is the sharing of resources without interference using mutual exclusion. The other purpose is the coordination of the process interactions in an operating system.

For example, when we start two or more threads within a program, there may be a situation when multiple threads try to access the same resource and then produce unforeseen result due to concurrency issues. If multiple threads try to write within a same file then they may corrupt the data because one of the threads can override data or while one thread is opening the same file at same time another thread might be closing the same file.
So there is need to synchronize the action of multiple threads and make sure only one thread can access the resource at a given point in time.

Answer to Question 3

```
abstract class ShapeFactory implements Shape{

    public static Shape getInstance (String s, double a){
        Circle cir = new Circle (a);
        return cir;
    }

    public static Shape getInstance (String s, int a, int b){
        Rectangle rect = new Rectangle (a, b);
        return rect;
    }

    public static Shape getInstance (String s, int a){
        Square sqr = new Square (a);
        return sqr;
    }
}
```

```
public class Rectangle extends ShapeDemo implements Shape {
    double b,h;
    public Rectangle (double b, double h){
        this.b = b;
        this.h = h;
    }

    @Override
    public double getArea(){
        double area = b * h;
        return area;
    }

    @Override
    public double getPerimeter(){
        double perimeter = 2 * (b+h);
        return perimeter;
    }

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

```
public class Square extends ShapeDemo implements Shape {
    double a;

    public Square (double a){
        this.a = a;
    }

    @Override
    public double getArea(){
        double area= a * a;
        return area;
    }

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

    @Override
    public double getPerimeter (){
        double perimeter = 4 * a;
        return perimeter;
    }
}
```

```
public class Circle extends ShapeDemo implements Shape{
    double r;

    public Circle (double r){
        this.r = r;
    }
    @Override
    public double getArea(){
        double area = Math.PI * r * r;
        return area;
    }

    @Override
    public double getPerimeter(){
        double perimeter = 2 * Math.PI * r;
        return perimeter;
    }

    @Override
    public String toString(){
        return "Circle";
    }
}
```