Assignment 3

1. Explain polymorphism.

Polymorphism is one of the core concepts of object-oriented programming (OOP) and describes situations in which something occurs in several different forms. In computer science, it describes the concept that you can access objects of different types through the same interface.

2. What is overloading?

Overloading occurs when two or more methods in one class have the same method name but different parameters. Overriding occurs when two methods have the same method name and parameters. One of the methods is in the parent class, and the other is in the child class.

3. What is overriding?

In any object-oriented programming language, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, same parameters or signature, and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.

4. What does the final mean in this method: public void doSomething(final Car aCar){}

Java always makes a copy of parameters before sending them to methods. This means the final doesn't mean any difference for the calling code. This only means that inside the method the variables can not be reassigned.

5. Suppose in question 4, the Car class has a method setColor(Color color){…}, inside doSomething method, Can we call aCar.setColor(red);?

Can't

6. Can we declare a static variable inside a method?

can't declare a static variable inside a method, static means that it's a variable/method of a class, it belongs to the whole class but not to one of its certain objects.

7. What is the difference between interface and abstract class?

An abstract class allows you to create functionality that subclasses can implement or override. An interface only allows you to define functionality, not implement it.

8. Can an abstract class be defined without any abstract methods?

yes, you can declare abstract class without defining an abstract method in it. Once you declare a class abstract it indicates that the class is incomplete and, you cannot instantiate it

9. Since there is no way to create an object of abstract class, what's the point of constructors of abstract class?

We need to initialize the non-abstract methods and instance variables, therefore abstract classes have a constructor.

10. What is a native method?

The use of native methods limits the portability of an application, because it involves system-specific code. Native methods can either be new native code statements or native code statements that call existing native code.

11. What is marker interface?

It is an empty interface (no field or methods). Examples of marker interface are Serializable, Cloneable and Remote interface. All these interfaces are empty interfaces.

12. Why to override equals and hashCode methods?

Collections such as HashMap and HashSet use a hashcode value of an object to determine how it should be stored inside a collection, and the hashcode is used again in order to locate the object in its collection.

Hashing retrieval is a two-step process:

Find the right bucket (using hashCode())

Search the bucket for the right element (using equals())

13. What's the difference beween int and Integer?

A Java both int and Integer are used to store integer type data the major difference between both is type of int is primitive while Integer is of class type.

14. What is serialization?

Serialization is the process of turning an object in memory into a stream of bytes so you can do stuff like store it on disk or send it over the network.

15. Create List and Map. List A contains 1,2,3,4,10(integer) . Map B contains ("a","1") ("b","2") ("c","10")   (key = string, value = string)
    Question: get a list which contains all the elements in list A, but not in map B.
16. Implement a group of classes that have common behavior/state as Shape. Create Circle, Rectangle and Square for now as later on we may need more shapes. They should have the ability to calculate the area. They should be able to compare using area. Please write a program to demonstrate the classes and comparison. You can use either abstract or interface. Comparator or Comparable interface.

```java
package Day4;

public class ShapeAssignment {
    public static void main(String[] args) {
        // Rectangle test
        double width = 5, length = 7;
        Shape rectangle = new Rectangle(width, length);
        System.out.println("Rectangle width: " + width + " and length: " + length
                + "\nResulting area: " + rectangle.area()
                + "\n");

        // Circle test
        double radius = 5;
        Shape circle = new Circle(radius);
        System.out.println("Circle radius: " + radius
                + "\nResulting Area: " + circle.area()
                + "\n");

        // S1uare test
        double a = 5;
        Shape square = new Square(a);
        System.out.println("Square sides lengths: " + a
                + "\nResulting Area: " + square.area()
                + "\n");
    }
}

abstract class Shape {
    public abstract double area();
}

class Rectangle extends Shape {
    private final double width, length; //sides

    public Rectangle() {
        this(1,1);
    }
    public Rectangle(double width, double length) {
        this.width = width;
        this.length = length;
    }

    @Override
    public double area() {
        // A = w * l
        return width * length;
    }

}

class Circle extends Shape {
    private final double radius;
    final double pi = Math.PI;

    public Circle() {
        this(1);
    }
    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double area() {
        // A = π r^2
        return pi * Math.pow(radius, 2);
    }
}

class Square extends Shape {
    private final double a; // sides

    public Square() {
        this(1);
    }
    public Square(double a) {
        this.a = a;
    }

    @Override
    public double area() {
        // Heron's formula:
        // A = SquareRoot(s * (s – a) * (s – b) * (s – c))
        // where s = (a + b + c) / 2, or 1/2 of the perimeter of the triangle
        return a * a;
    }
}
```