Assignment 3

1. Explain polymorphism.
2. What is overloading?
3. What is overriding?
4. What does the final mean in this method: public void doSomething(**final** Car aCar){}
5. Suppose in question 4, the Car class has a method setColor(Color color){…}, inside doSomething method, Can we call aCar.setColor(red);?
6. Can we declare a static variable inside a method?
7. What is the difference between interface and abstract class?
8. Can an abstract class be defined without any abstract methods?
9. Since there is no way to create an object of abstract class, what's the point of constructors of abstract class?
10. What is a native method?
11. What is marker interface?
12. Why to override equals and hashCode methods?
13. What's the difference beween int and Integer?
14. What is serialization?
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.

1. Explain polymorphism.
   Polymorphism allows us to perform a single action in different ways.
2. What is overloading?
   Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters or both.
3. What is overriding?
   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.
4. What does the final mean in this method: public void doSomething(**final** Car aCar){}
   Cannot change the value of aCar.
5. Suppose in question 4, the Car class has a method setColor(Color color){…}, inside doSomething method, Can we call aCar.setColor(red);?

6. Can we declare a static variable inside a method?
   No. Static Variables are allocated memory at class loading time because they are part of the class and not its object. 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?
   Interface can have only abstract methods. An abstract class can have abstract and non-abstract methods, it can have default and static methods also. Abstract class can have final, non-final, static, and non-static variables. The interface has only static and final variables. An interface can extend another Java interface only, an abstract class can extend another Java class and implement multiple Java interfaces. Members of a Java interface are public by default. A Java abstract class can have class members like private, protected, etc.
8. Can an abstract class be defined without any abstract methods?
   Yes.
9. Since there is no way to create an object of abstract class, what's the point of constructors of abstract class?
   An abstract class's constructor being called doesn't mean that its object is created. Abstract class constructors will be called when its concrete subclass will be instantiated
10. What is a native method?
    Native methods are Java methods that start in a language other than Java. Native methods can access system-specific functions and APIs that are not available directly in Java.
11. What is marker interface?
    A marker interface is an interface that has no methods or constants inside it. It provides run-time type information about objects, so the compiler and JVM have additional information about the object.
12. Why to override equals and hashCode methods?

Failure to do so will result in a violation of the general contract for Object.hashCode(), which will prevent your class from functioning properly in conjunction with all hash-based collections, including HashMap, HashSet, and Hashtable.

If only equals is overriden, then when you call myMap.put(new String("s"),someValue) first will hash to some bucket and when you call myMap.put(new String("s"),someOtherValue) it will hash to some other bucket (as they have a different hashCode). So, although they are equal, as they don't hash to the same bucket, the map can't realize it and both stay in the map.

If you only override hashCode then when you call myMap.put(new String("s"),someValue) it takes first, calculates its hashCode and stores it in a given bucket. Then when you call myMap.put(new String("s"),someOtherValue) it should replace first with second as per the Map Documentation because they are equal (according to the business requirement).

But the problem is that equals was not redefined, so when the map hashes second and iterates through the bucket looking if there is an object k such that second.equals(k) is true it won't find any as second.equals(first) will be false.

13. What's the difference beween int and Integer?
    int is a data type that stores 32 bit signed two's compliment integer. On other hand Integer is a wrapper class which wraps a primitive type int into an object.
14. What is serialization?
    Serialization is a mechanism of converting the state of an object into a byte stream.
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.

```
public static void q1(){
    List<Integer> list = new ArrayList<>();
    list.add(1);
    list.add(2);
    list.add(3);
    list.add(4);
    list.add(10);
    Map<String,String> map = new HashMap<>();
    map.put("a","1");
    map.put("b","2");
    map.put("c","10");

    List<Integer> res = new ArrayList<>();
    for(int i : list){
        if(!map.containsValue(i + "")){
            res.add(i);
        }
    }
    System.out.println(res);
}
```

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.

```
public static void q2(){
        Circle circle = new Circle(1.0);
        Square square = new Square(2.0);
        Rectangle rectangle = new Rectangle(1,2);
        List<Shape> shapes = new ArrayList<>();
        shapes.add(circle);
        shapes.add(square);
        shapes.add(rectangle);
        Collections.sort(shapes, new Comparator<Shape>() {
            @Override
            public int compare(Shape o1, Shape o2) {
                return Double.compare(o1.calculate(),o2.calculate());
            }
        });
        System.out.println(shapes);
    }
}

abstract class Shape{
    abstract double calculate();

}

class Circle extends Shape{
    double radius;

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

    }

    @Override
    double calculate() {
        return 3.14 * Math.pow(radius,2);
    }

    @Override
    public String toString() {
        return "Circle(" + radius + "):" + calculate();
    }
}

class Rectangle extends Shape{
    double length;
    double width;

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

```
    }

    @Override
    double calculate() {
        return length * width;
    }

    @Override
    public String toString() {
        return "Rectangle(" + length + " " + width +"):" + calculate();
    }
}

class Square extends Shape{

    double side;

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

    @Override
    double calculate() {
        return Math.pow(side,2);
    }

    @Override
    public String toString() {
        return "Square(" + side + "):" + calculate();
    }

}
```