Assignment 3

1. **Explain polymorphism.**
   Polymorphism means “many forms” and in java it refers to the process of inheritance and using the methods provided by the inherited classes for different actions.
2. **What is overloading?**
   Overloading is when you have the same method but with different parameters. For example you could have a default constructor that takes in no parameters and a constructor method that allows certain parameters to set the default values for the object of that class.
3. **What is overriding?**
   Overriding is when you override an inherited method meaning you are changing or filling in the body the method that was inherited from the super class.
4. **What does the final mean in this method: public void doSomething(final Car aCar){}**
   Final means that we can’t modify the object aCar of class Car that is being passed in as the parameter for the method doSomething. This value, or in this case the object aCar, can not be modified.
5. **Suppose in question 4, the Car class has a method setColor(Color color){…}, inside doSomething method, Can we call aCar.setColor(red);?**
   No. You would need to access the inner method by its own name.
6. **Can we declare a static variable inside a method?**
   No.
7. **What is the difference between interface and abstract class?**
   A class can implement multiple interfaces, a class can only extend one abstract class at a time.
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 can still have non abstract attributes and methods that can be initialized.
10. **What is a native method?**
    Native methods start in another language.
11. **What is marker interface?**
    Basically an empty interface, also known as a tagging interface.
12. **Why to override equals and hashCode methods?**
    We might want to override equals to change how an object is compared when sorting for instance. If we override equals, then we must override hashCode as well because of their unique relationship. The equals method is actually checking if the references to memory location are the same when it compares two values/objects which relates to the hash value of those objects.
13. **What’s the difference beween int and Integer?**
    int is a primitive data type and Integer is a reference type and a wrapper class for int. An Integer object can have methods.
14. **What is serialization?**

Serialization is the process of turning an object into a stream of bytes.

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.

```
Map<String, Integer> map = new HashMap<>(); // create hashmap and fill it with values

                map.put("a", 1);
                map.put("b", 2);
                map.put("c", 10);

                List<Integer> list = new ArrayList<>(); // create array list and fill
it with values

                list.add(1);
                list.add(2);
                list.add(3);
                list.add(4);
                list.add(10);

                List<Integer> results = new ArrayList<>(); // create array list for
results

                for (int i : list) { // iterate through original array list and check
if the hashmap contains the value, if not add it to the results list
                    if (!map.containsValue(i)) results.add(i);
                }

                        System.out.println(results);
```

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 abstract class Shape implements Comparator<Shape> {
                public abstract double area(); // area returns double because circle
returns double
                public int compare(Shape shape) {
                    return (int) (this.area() - shape.area());
                }
            }

               public class Circle extends Shape implements Comparator<Shape>{
                      private double radius;

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

                      public double area() {
                          return Math.PI * (radius * radius);
                      }
```

```
            public int compare(Shape shape) {
                return (int) (this.area() - shape.area());
            }
        }


        public class Square extends Shape implements Comparator<Shape>{
            private int edgelen;
            public Square(int a) {
                this.edgelen = a;
            }


            public double area() {
               return edgelen * edgelen;
            }

            public int compare(Shape shape) {
                return (int) (this.area() - shape.area());
            }
        }


public class Rectangle extends Shape implements Comparator<Shape>{
    private int width, length;

    public Rectangle(int w, int l) {
        this.width = w;
        this.length = l;
    }

    public double area() {
        return width * length;
    }

    public int compare(Shape shape) {
        return (int) (this.area() - shape.area());
    }
}
```