Assignment 3

1. Explain polymorphism.

   It often appears in two situation, one is override a method, the other is overloading. When makes function call, the real implementation depends on the overriding and overloading method

2. What is overloading?

   For example, two functions:

   Public void A(String s) {}

   Public void A(int i) {}

   Two methods are in the same class but their parameters are different

3. What is overriding?

   It happens when modify the method of parent class or parent interfaces in current class

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

   It means the value of aCar can't be changed anymore. Which means aCar always refer to this constant Car object

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

   Yes, we can, if the setColor method is visible(not private or some other modifier)

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

   No, we can't. Since when we call a method, all its variables are stored on the call stack, it can't belong to a class

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

   Abstract class can have constructor, implemented method and non-public method

   Interface only have public abstract method

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

   Yes, we can. Abstract class means it possibly has some abstract method, not necessary

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

   It is used for its subclass which extends that abstract class

10. What is a native method?

    Native method can't be implemented in java, actually this is a method to call non-java program

11. What is marker interface?

    There is no constant variable or method in it, for example interface serializable is one of the marker interfaces

12. Why to override equals and hashCode methods?

    Since equals method is to compare the content between two objects. If we override the equals method, that means we define a rule to compare two objects. But when we want to use HashMap/HashSet data structure, we have to keep two different objects have the same hash code when the two objects are equal

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

Int is primitive type, but Integer is its wrapper class. There are many methods in Integer

14. What is serialization?

Serialization means we can convert the fields of one object to byte stream, this is helpful when we transfer information between two servers

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 class Fifteen {
    public static void main(String[] args) {
        Fifteen f = new Fifteen();
        Map<String, String> m = new HashMap<>();
        m.put("a", "1");
        m.put("b","2");
        m.put("c","10");
        List<Integer> l = Arrays.asList(1,2,3,4,10);
        l = f.dedup(m, l);
        System.out.println(l);
    }
    private List<Integer> dedup(Map<String, String> map, List<Integer> l) {
        Set<Integer> set = new HashSet<>();
        for (Map.Entry<String, String> e : map.entrySet()) {
            Integer i = Integer.valueOf(e.getValue());
            set.add(i);
        }
        List<Integer> res = new ArrayList<>();
        for (Integer i : l) {
            if (!set.contains(i)) {
                res.add(i);
            }
        }
        return 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 abstract class Shape implements Comparable<Shape>{
    public int area;
    public abstract int compareTo(Shape s);
    public abstract int calculate();
}
class Circle extends Shape {
    int radius;
    public int compareTo(Shape s) {
        Circle c = (Circle) s;
        return area < c.area ? -1 : 1;
    }
    public int calculate() {
        return 3 * radius * radius;
    }
    public Circle(int radius) {
        super();
        this.radius = radius;
        this.area = calculate();
    }
}
class Rectangle extends Shape {
    int width;
    int length;
    public int compareTo(Shape s) {
        Circle c = (Circle) s;
        return area < c.area ? -1 : 1;
    }
    public int calculate() {
        return width * length;
    }
    public Rectangle(int width, int length) {
        super();
        this.width = width;
        this.length = length;
        this.area = calculate();
    }
}
class Square extends Shape {
```

```
    int width;
    public int compareTo(Shape s) {
        Circle c = (Circle) s;
        return area < c.area ? -1 : 1;
    }
    public int calculate() {
        return width * width;
    }
    public Square(int width) {
        super();
        this.width = width;
        this.area = calculate();
    }
}
```