Flowcharts [15 points]

Draw a **flowchart** that receives some positive integer numbers and calculates and prints how many *odd* and how many *even* numbers it has received. The program stops, when it receives a non positive number.

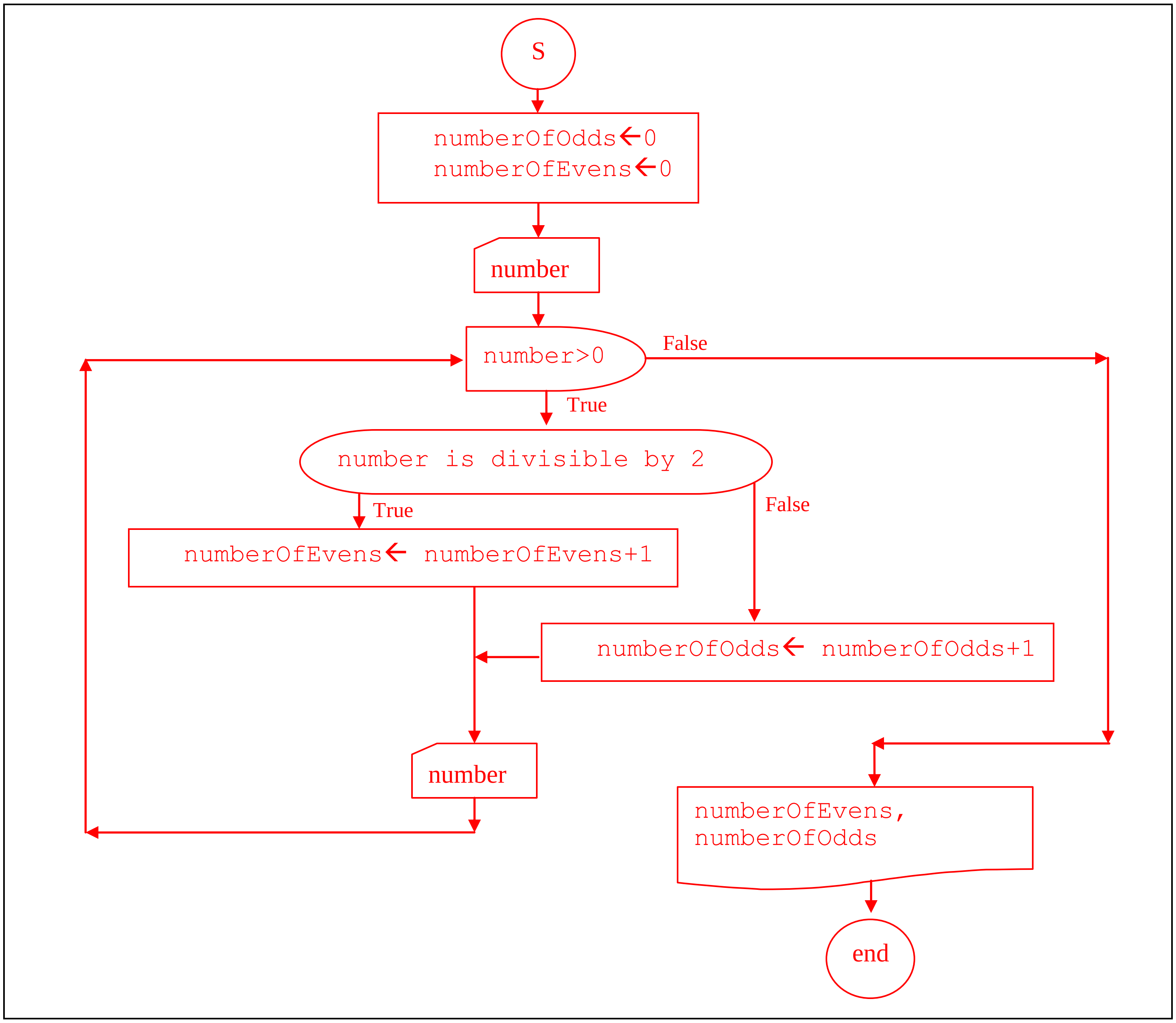
Recall: An *even* number is a number that is divisible by 2 (the remainder of the division is 0); otherwise, it is an *odd* number.

For example: If it receives the following numbers: `5 3 4 1 20 15 8 5 10 3 -11`

it should print: `There are 6 odd numbers and 4 even numbers.`

As another example, if the program receives the following number: `111 8 3 100 2 24 0`

it should print: `There are 2 odd numbers and 4 even numbers.`



Java Class [25 points]

Write a program in Java that defines a new class of numbers called `SpecialNum`. Each object of this class has an integer value. Furthermore, this class has three methods, namely *`reverse`*, *`autoReverse`*, and *`getValue`*.

Method *`reverse`* returns the reverse of the integer value of the object. For example, one may invoke method `reverse` as follows:

```
int x,y;
…
SpecialNum obj1=new SpecialNum(x);
y = obj1.reverse();
```

Method `autoReverse` determines if an object of class SpecialNum is auto-reversal or not—by returning either **true** or **false**. For example, one may invoke method `autoReverse` as follows:

```
int x;
…
SpecialNum obj1=new SpecialNum(x);
if (obj1.autoReverse())
     System.out.println("yes "+obj1.getValue()+" is auto-reverse")
else System.out.println("no "+obj1.getValue()+" is not auto-reverse");
```

Note: Provide appropriate documentation.

```
public class SpecialNum {
          /**
      * Constructs a special number that its integer value is
      * @param x
      */
     public SpecialNum (int x){
          this.x=x;
     }
     /**
      * returns the reverse of an integer value
      * @return
      */
     public int reverse(){
          int x=0, temp=this.x;
          while (temp>0){
               x=x*10+ (temp % 10);
               temp=temp / 10;
          }
          return x;
     }
```

```
	/**
	 * if the integer value of an object is equal to its reverse
	 * autoreverse returns TRUE
	 * @return
	 */
	public boolean autoReverse(){

		if (x==this.reverse())
			return true;
		else return false;
	}
	/**
	 * rtuens the integer value of a special number
	 * @return
	 */
	public int getValue(){
		return x;
	}
	/**
	 * the integer value of the object
	 */
	private int x;
}
```

True/False Questions [65 Points]

True Statements:

1. An array is a sequence of values of the same type.

True

2. A limitation of arrays is that they have fixed length.

True

3. The last element in the array has an index one less than the array length.

True

4. An ArrayList cannot store primitive data types

True

5. The ArrayList class is a generic class

True

6. An arrays is usually partially filled.

True

7. The following declaration is true.

```
int valuesLength = in.nextInt();
double[] values = new double[valuesLength];
```

True

8. A class should represent a single concept from a problem domain.

True

9. A method can never change parameters of primitive data types.

True

10. A static variable belongs to the class, not to any particular object of the class.

True

11. The substring method of the String class is an accessor method.

True

12. The JUnit philosophy is to run all tests every time you change your code.

True

13. A precondition is a requirement that the caller of a method must meet.

True

14. We don't need to import other classes in the same package

True

15. It's not a good practice to minimize the cohesion between classes.

True

FALSE Statement:

16. The scope of a local variable can contain the declaration of another local variable with the same name.

False

17. A && B is the same as B && A for any Boolean condition A and B.

False

18. *values[1]* is the first element of array *values[ ]*.

False

19. An advantage of ArrayLists is that they have fixed length.

False

20. Length() is a method that returns size of an array

False

21. An ArrayList is a sequence of values of different types.

False

22. There are 32 wrapper class in Java

False

23. Java does not support 4 Dimensional arrays.

False

24. In most cases, it's better to use parallel arrays instead of arrays of objects.

False

25. Regression Testing means to perform both black-box and white box testings**.**

False

26. In an enhanced for loop, the loop variable contains an index not an element.

False

27. In ArrayLists, setting an element into an empty position returns error.

False

28. The following for each is not legal:

for (double element : values){element = 0;}

False

29. The following declaration is true.

double[] values = new int[10]

False

30. The following declaration is true. *Array value = new array(int, 10);*

False

31. *for (int i = 0; i < 10; ++i) {do P}* skips the statement P for i=0.

False

32. *ArrayList<int> name = new ArrayList<int>();* restores a list of integer numbers.

False

33. *int i = names.size(); name = names.get(i);* returns the last element in the arraylist name.

False

34. *ArrayList<Integer> test = new ArrayList<Integer>();* *int i=20;* *test.add(i);* returns [20].

False

35. An immutable class has mutator methods.

False

36. It is not a good practice to minimize the coupling between classes.

False

37. References to objects of an immutable class cannot be safely shared.

False

38. Side effects are 100% avoidable.

False

39. A static method is invoked on an object.

False

40. You should give each variable the biggest scope that it could have.

False

41. A local variable cannot shadow an instance variable with the same name.

False

42. If the caller does not meet the precondition of our method, w have to notify them

False

43. Static constants should always be declared as private

False

44. A side effect of a method is any kind of modification of data that is observable inside the method.

False

45. We always need to import java.lang

False