QUESTION 1:

This question involves the `WordMatch` class, which stores a secret string and provides methods that compare other strings to the secret string. You will write two methods in the `WordMatch` class.

```
public class WordMatch {

        /**
         * Constructs a WordMatch object with the given secret string of lowercase
         * letters.
         */
        public WordMatch(String word) {
                /* to be implemented */
                ...
        }
        /**
         * Returns a score for guess, as described.
         * Precondition: 0 <guess.length() <= secret.length()
         */
        public int scoreGuess(String guess) {
                /* to be implemented */
                ...
       }

        /**
         * Returns the better of two guesses, as determined.
         * Precondition: guess1 and guess2 contain all lowercase letters.
         * guess1 is not the same as guess2.
         */
        public String findBetterGuess(String guess1, String guess2) {
                /* to be implemented */
         ...
         }
}
```

scoreGuess method:

To determine the score to be returned, `scoreGuess` finds the number of times that `guess` occurs as a substring of `secret` and then multiplies that number by the square of the length of `guess`. Occurrences of `guess` may overlap within `secret`. Assume that the length of `guess` is less than or equal to the length of `secret` and that `guess` is not an empty string.

```
WordMatch game = new WordMatch("mississippi");
```

Value of `guess`	Number of Substring Occurrences	Score Calculation: (Number of Substring Occurrences) x (Square of the Length of `guess`)	Return Value of `game.scoreGuess(guess)`
`"i"`	4	`4 * 1 * 1 = 4`	4
`"iss"`	2	`2 * 3 * 3 = 18`	18
`"issipp"`	1	`1 * 6 * 6 = 36`	36
`"mississippi"`	1	`1 * 11 * 11 = 121`	121

```
WordMatch game = new WordMatch("aaaabb");
```

Value of `guess`	Number of Substring Occurrences	Score Calculation: (Number of Substring Occurrences) x (Square of the Length of `guess`)	Return Value of `game.scoreGuess(guess)`
`"a"`	4	`4 * 1 * 1 = 4`	4
`"aa"`	3	`3 * 2 * 2 = 12`	12
`"aaa"`	2	`2 * 3 * 3 = 18`	18
`"aabb"`	1	`1 * 4 * 4 = 16`	16
`"c"`	0	`0 * 1 * 1 = 0`	0

findBetterGuess method:

Returns the better guess of its two `String` parameters, `guess1` and `guess2`. If the `scoreGuess` method returns different values for `guess1` and `guess2`, then the guess with the higher score is returned. If the `scoreGuess` method returns the same value for `guess1` and `guess2`, then the alphabetically greater guess is returned.
The following example shows a declaration of a `WordMatch` object and the outcomes of some possible calls to the `scoreGuess` and `findBetterGuess` methods.

```
WordMatch game = new WordMatch("concatenation");
```

Method Call	Return Value	Explanation
`game.scoreGuess("ten");`	9	`1 * 3 * 3`
`game.scoreGuess("nation");`	36	`1 * 6 * 6`
`game.findBetterGuess("ten", "nation");`	`"nation"`	Since `scoreGuess` returns 36 for `"nation"` and 9 for `"ten"`, the guess with the greater score, `"nation"`, is returned.
`game.scoreGuess("con");`	9	`1 * 3 * 3`
`game.scoreGuess("cat");`	9	`1 * 3 * 3`
`game.findBetterGuess("con", "cat");`	`"con"`	Since `scoreGuess` returns 9 for both `"con"` and `"cat"`, the alphabetically greater guess, `"con"`, is returned.

QUESTION 2:

Create a class called 'Matrix' containing constructor that initializes the number of rows and number of columns of a new Matrix object.

The Matrix class has the following information:

1. number of rows of matrix
2. number of columns of matrix
3. elements of matrix in the form of 2D array

The Matrix class has methods for each of the following:

1. get the number of rows
2. get the number of columns
3. set the elements of the matrix at given position (i,j)
4. adding two matrices. If the matrices are not addable, "Matrices cannot be added" will be displayed.
5. multiplying the two matrices

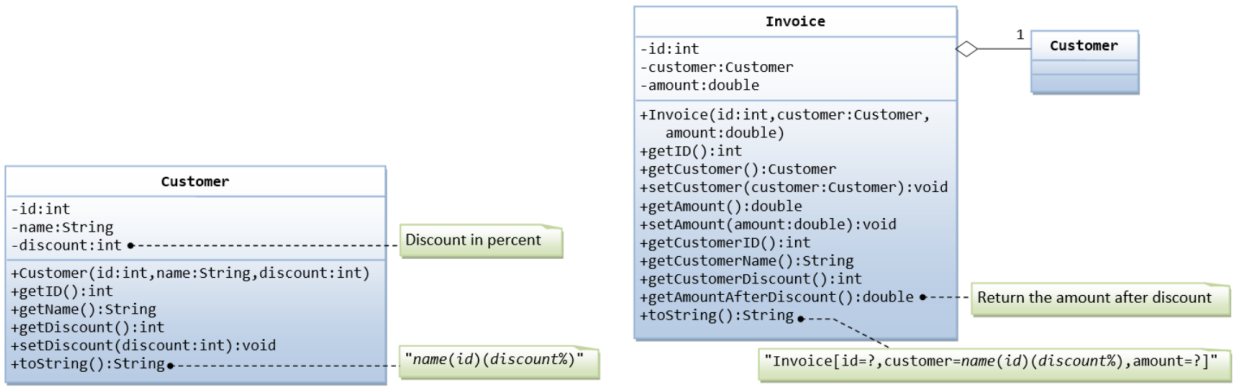
QUESTION 3:

Write a program by creating an 'Employee' class having the following methods and print the final salary.

1. 'getInfo()' which takes the salary, number of hours of work per day of employee as parameter
2. 'addSal()' which adds $10 to salary of the employee if it is less than $500.
3. 'addWork()' which adds $5 to salary of employee if the number of hours of work per day is more than 6 hours

QUESTION 4:

A class called Customer, which models a customer in a transaction, is designed as shown in the class diagram. A class called Invoice, which models an invoice for a particular customer and composes an instance of Customer as its instance variable, is also shown. Write the Customer and Invoice classes.



Below is a test class:

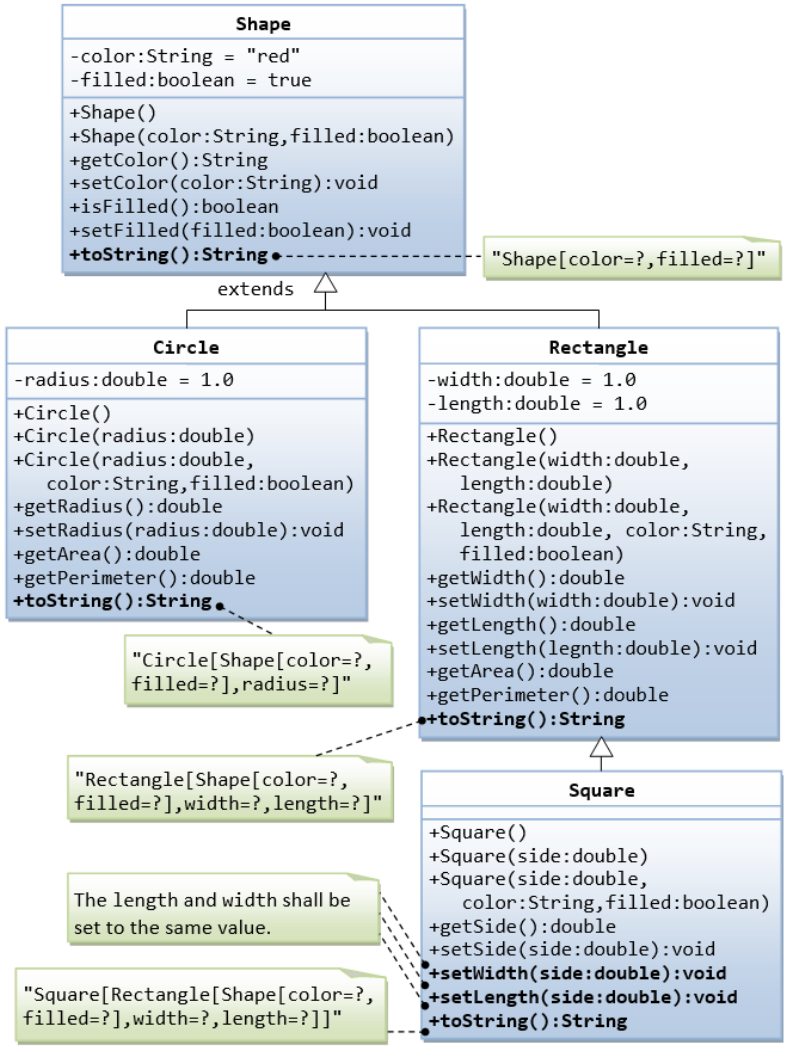
```
public class TestMain {

   public static void main(String[] args) {

      // Test Customer class
      Customer c1 = new Customer(88, "Tan Ah Teck", 10);
      System.out.println(c1);  // Customer's toString()

      c1.setDiscount(8);
      System.out.println(c1);
      System.out.println("id is: " + c1.getID());
      System.out.println("name is: " + c1.getName());
      System.out.println("discount is: " + c1.getDiscount());

      // Test Invoice class
      Invoice inv1 = new Invoice(101, c1, 888.8);
      System.out.println(inv1);

      inv1.setAmount(999.9);
```

```
      System.out.println(inv1);
      System.out.println("id is: " + inv1.getID());
      System.out.println("customer is: " + inv1.getCustomer());  // Customer's toString()
      System.out.println("amount is: " + inv1.getAmount());
      System.out.println("customer's id is: " + inv1.getCustomerID());
      System.out.println("customer's name is: " + inv1.getCustomerName());
      System.out.println("customer's discount is: " + inv1.getCustomerDiscount());
      System.out.printf("amount after discount is: %.2f%n", inv1.getAmountAfterDiscount());
   }
}
```

The expected output is:

```
Tan Ah Teck(88)(10%)
Tan Ah Teck(88)(8%)
id is: 88
name is: Tan Ah Teck
discount is: 8
Invoice[id=101,customer=Tan Ah Teck(88)(8%),amount=888.8]
Invoice[id=101,customer=Tan Ah Teck(88)(8%),amount=999.9]
id is: 101
customer is: Tan Ah Teck(88)(8%)
amount is: 999.9
customer's id is: 88
customer's name is: Tan Ah Teck
customer's discount is: 8
amount after discount is: 919.91
```

QUESTION 5:



Write a superclass called `Shape` (as shown in the class diagram), which contains:

- Two instance variables `color` (`String`) and `filled` (`boolean`).
- Two constructors: a no-arg (no-argument) constructor that initializes the `color` to "green" and `filled` to `true`, and a constructor that initializes the `color` and `filled` to the given values.
- Getter and setter for all the instance variables. By convention, the getter for a `boolean` variable `xxx` is called `isXXX()` (instead of `getXxx()` for all the other types).
- A `toString()` method that returns "`A Shape with color of xxx and filled/Not filled`".

Write a test program to test all the methods defined in `Shape`.
Write two subclasses of `Shape` called `Circle` and `Rectangle`, as shown in the class diagram.

The `Circle` class contains:

- An instance variable `radius` (`double`).
- Three constructors as shown. The no-arg constructor initializes the radius to `1.0`.

- Getter and setter for the instance variable `radius`.
- Methods `getArea()` and `getPerimeter()`.
- Override the `toString()` method inherited, to return "`A Circle with radius=xxx, which is a subclass of yyy`", where `yyy` is the output of the `toString()` method from the superclass.

The `Rectangle` class contains:

- Two instance variables `width` (`double`) and `length` (`double`).
- Three constructors as shown. The no-arg constructor initializes the `width` and `length` to `1.0`.
- Getter and setter for all the instance variables.
- Methods `getArea()` and `getPerimeter()`.
- Override the `toString()` method inherited, to return "`A Rectangle with width=xxx and length=zzz, which is a subclass of yyy`", where `yyy` is the output of the `toString()` method from the superclass.

Write a class called `Square`, as a subclass of `Rectangle`. Convince yourself that `Square` can be modeled as a subclass of `Rectangle`. `Square` has no instance variable, but inherits the instance variables width and length from its superclass Rectangle.

- Provide the appropriate constructors (as shown in the class diagram). Hint:

```
public Square(double side) {
   super(side, side);  // Call superclass Rectangle(double, double)
}
```

- Override the `toString()` method to return "`A Square with side=xxx, which is a subclass of yyy`", where `yyy` is the output of the `toString()` method from the superclass.
- Do you need to override the `getArea()` and `getPerimeter()`? Try them out.
- Override the `setLength()` and `setWidth()` to change both the `width` and `length`, so as to maintain the square geometry.