Problem 1: Scoping (10 points)

Read the following Java excerpt.

```
public class Problem1
{
   public static int m = 0;

   public static void main(String args[])
   {
      int i = 0;
      for (int j = 1; j <= 3; j++)
      {
         doStuff(5);
      }
      System.out.println("The value of i is " + i + ".");
      System.out.println("The value of m is " + m + ".");
   }

   public static int doStuff(int j)
   {
      int k = 0, i = 0;
      while (k < j)
      {
         i++;
         m++;
         k++;
      }
      System.out.println("Inside doStuff: the value of i is " + i + ".");
      return i/2;
   }
}
```

What is the output of the above code? (fill in the blanks)

Inside doStuff: the value of i is _____5_____.

Inside doStuff: the value of i is _____5_____.

Inside doStuff: the value of i is _____5_____.

The value of i is _____0_____.

The value of m is _____15_____.

Each blank is worth two points.

Problem 2: Object-oriented Programming

Part A (10 points): Design Questions

Provide a short answer to each of the following questions. "Short answer" means 1-3 sentences.

Each answer is worth two points.

1. According to what's been said in lecture or by Savitch, why should instance variables be declared private?

   (2 points) Here are two sample answers:

   Protection: other programmers cannot modify the instance variables if they are private

   Abstraction/encapsulation: "private" allows you to hide the details of the implementation

2. What is a default constructor? (How do you know that a constructor is a default constructor, rather than some other kind of constructor? What does a default constructor do?)

   (1 point) Default constructors are distinguished in that they take in no arguments.

   (1 point) A default constructor provides default values for all instance variables.

3. What is the difference between a class and an object?

   (1 point) A class is a pattern/template/abstract description.

   (1 point) An object is an instantiation of the pattern; it represents a specific item of that class type.

4. What does "method overloading" mean?

   (1 point) Two or more methods that share the same name.

   (1 point) The methods are distinguished in that they take in different arguments.

5. If a method is **not** declared static, can it call other "static" methods? Why or why not?

   (1 point) Yes.

   (1 point) Any object has access to all class methods, so a non-static (object) method can call a static (class) method.

Part B (5 points correctness, 5 points style): Constructors

Read the following class description and **implement the default constructor**. Then, provide an **additional constructor** of your choice. It should take in at least one argument and do something reasonable (for example, it should use each input argument somewhere in the constructor's body).

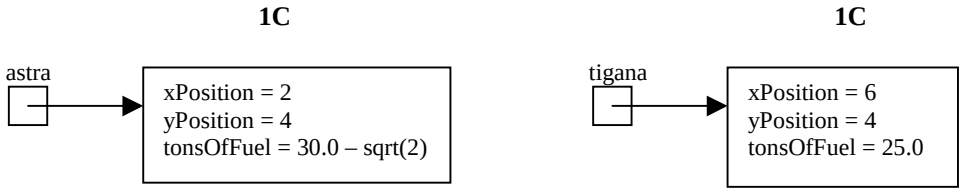
```
public class Clock
{
   int hour;
   int minute;
   boolean AM; // true if the time is before noon;
               // false otherwise (noon is 12:00 p.m.; midnight is 12 a.m.)

   // default constructor: set Clock to midnight
   public Clock()
   {
      hour = 12;
      minute = 0;
      AM = true;
   }
   // Correctness:
   // 2 total: 1 for hour and minute, 1 for AM
   // Style:
   // 1 total: indenting?

   // constructor: add your constructor here (don't forget comments!)

   // Any "reasonable" constructor is okay.
   // One example:

   // Constructor: takes in hour, minute, and AM information
   public Clock(int h, int m, boolean isAM)
   {
      hour = h;
      minute = m;
      AM = isAM;
   }

   // Correctness:
   // 3 total: 1 for method declaration, 1 for hour/minute, 1 for AM
   // Style:
   // 4 total: 2 for method comment, 2 for variable name choice

}
```

Part C (10 points): References

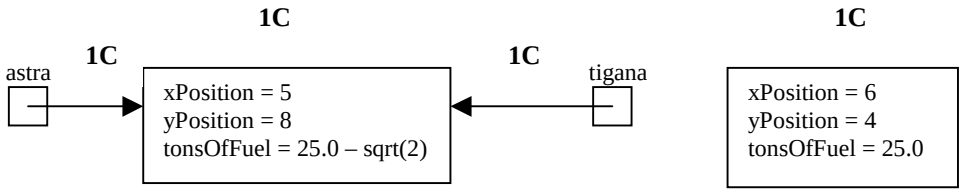
Read the following code and fill in the blanks with box-and-arrow pictures that indicate what the current **state** of the references (astra and tigana) are. As an example, the first one is done for you. This code is using the original Spaceship class, where the Spaceship's position is represented by two integers (xPosition and yPosition).

```
Spaceship astra = new Spaceship(3, 3);
Spaceship tigana = new Spaceship(6, 6);
```

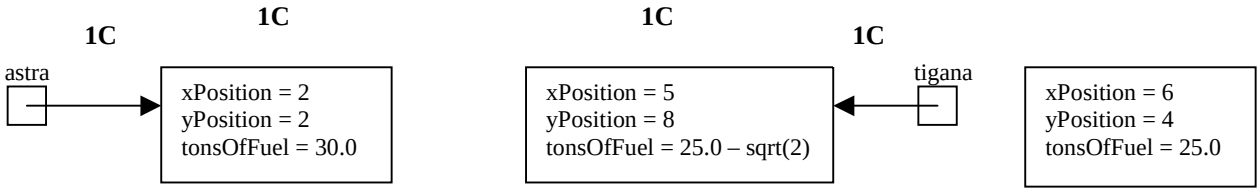

```
astra.move(-1, 1);
tigana.hyperjump(6, 4);
```




```
tigana = astra;
tigana.move(3, 4);
```



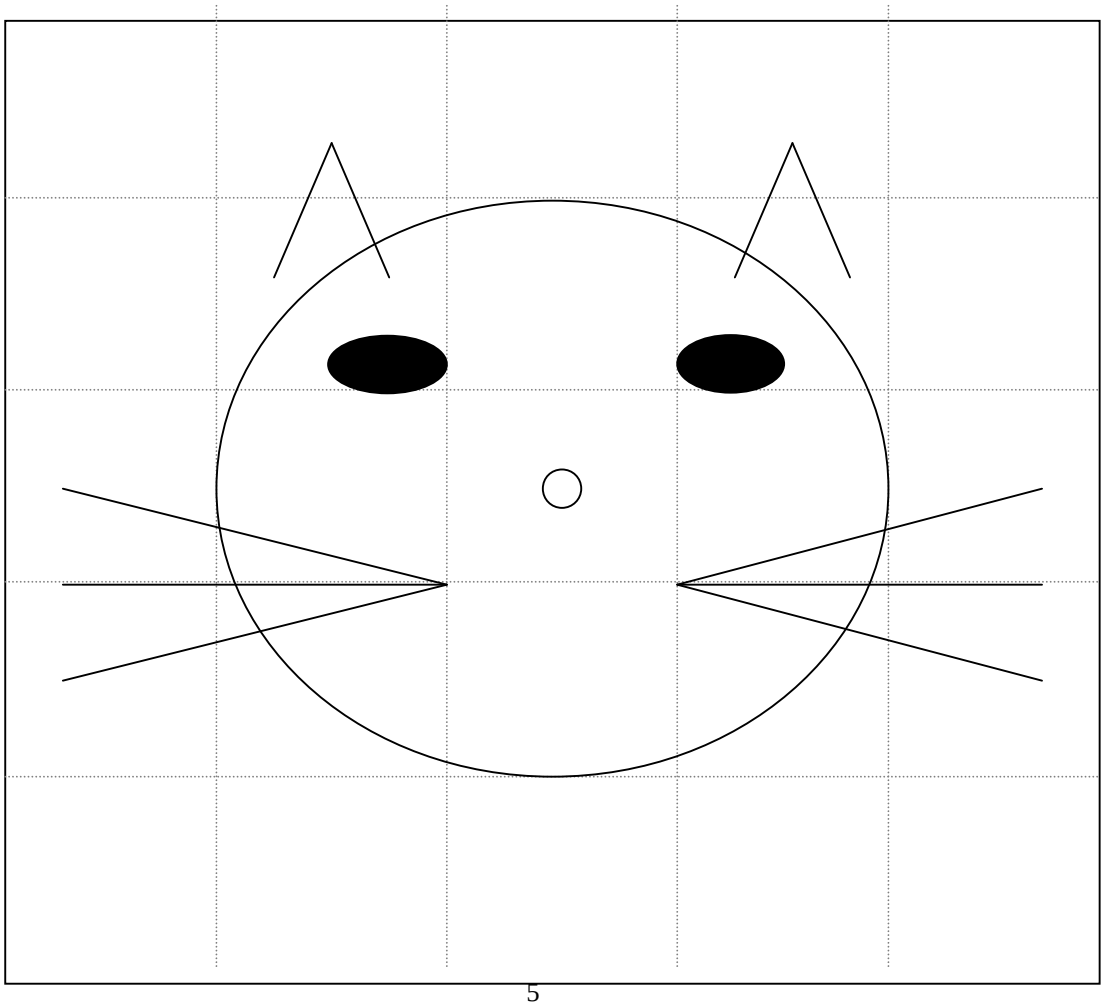

```
astra.hyperjump(12, 5);
astra = new Spaceship(2, 2);
```



Problem 3: Graphics (10 points)

Read through the following paint() method and draw the resulting output in the (500 by 500) box. You may find it helpful to label the dotted lines with numbers.

```
public void paint(Graphics g)
{
  g.drawOval(100, 100, 300, 300);    // 1C
  g.fillOval(150, 180, 50, 20); // 1C
  g.fillOval(300, 180, 50, 20); // 1C
  g.drawOval(240, 240, 20, 20); // 1C

  g.drawLine(150, 80, 170, 150); // 1C
  g.drawLine(150, 80, 130, 150);
  g.drawLine(350, 80, 370, 150); // 1C
  g.drawLine(350, 80, 330, 150);

  g.drawArc(150, 250, 100, 100, 0, -90); // 1C
  g.drawArc(250, 250, 100, 100, -90, -90); // 1C

  g.drawLine(20, 250, 200, 300); // 1C
  g.drawLine(20, 300, 200, 300);
  g.drawLine(20, 350, 200, 300);
  g.drawLine(300, 300, 480, 250); // 1C
  g.drawLine(300, 300, 480, 300);
  g.drawLine(300, 300, 480, 350);
} // end paint()
```



Problem 4: Arrays (10 points correctness, 10 points style)

Read through the following piece of code.

```
public class Problem4
{
   public static int[] myArray = new int[10];

   public static void main(String args[])
   {
      for (int i=0; i<myArray.length; i++)
      {
         myArray[i] = i + 1;
      }

      doStuff();
   }
}
```

Now, implement a **doStuff()** method for the Problem4 class (it is being called inside main()). doStuff() takes in no arguments and doesn't return anything. The action it takes is to replace each odd integer in **myArray** with twice that odd integer. Remember to **comment** the method properly. **Style** counts!

```
// doStuff: replaces all odd integers in myArray with their doubles
// input: none
// output: none
public static void doStuff()
{
   // Loop over the array
   for (int i=0; i<myArray.length; i++)
   {
     // Check if the entry is odd
     if (myArray[i]%2 == 1)
     {
        // If so, multiply it by two
        myArray[i] *= 2;
     }
   }
}
```

Correctness:
3C: method declaration (1 for "static", 1 for "void", 1 for empty arguments)
2C: loop (1 for setup and increment, 1 for condition)
3C: check if the entry is odd (1 for array indexing, 1 for %2, 1 for comparing to 1)
2C: array update (1 for LHS, 1 for RHS)

Style:
3S: method comment
2S: loop comment
1S: if comment
1S: update comment
1S: comparing to myArray.length instead of 10
1S: using a for loop (instead of a while loop or somethin)
1S: indenting