Answer all of the following questions. READ EACH QUESTION CAREFULLY. Answer each question in the space provided on these pages. Budget your time so you spend enough on the longer programming question at the end. There are a total of 50 points.

Keep your answers short and to the point. Good luck.

Question 1. (3 points) Complete the picture below to show the values that are bound to the following names after executing the following statements. The first one has been done for you.

```
int num = 17;
int another = num % 3 + 2;
int more = (num + 2) * 2;
int lastone = 2 * (num / 2);
```

num ⟶ 17

another ⟶ 4

more ⟶ 38

lastone ⟶ 16

Question 2. (8 points) For each of the following, indicate whether or not there are any errors in the code. If something is wrong, circle the error and indicate whether it is a syntax or semantic error.

(a) `int = 17;` **syntax error (missing name)**

(b) `int n = 42;` **OK**

(c) `int a = "hello";` **semantic error (type mismatch)**

(d) `String s = "howdy";` **OK**

(e) `String t = 17;` **semantic error (type mismatch)**

(f) `String u = "17";` **OK**

(g) `double pi 3.14159;` **syntax error (missing =)**

(h) `int bugnum = 100000+;` **syntax error (extra + or missing right operand of +)**

Question 3. (5 points) In lecture we talked about what happens when a method call is executed. Here are the steps, but they are not in the correct order.

Indicate the correct order in which these steps occur by writing numbers from 1 to 5 in the blank spaces in front of each step.

__4___ The method body's statements are evaluated until they are no more, or until a `return` statement is executed.

__2___ The method's parameter names are bound to the corresponding values of the arguments.

__5___ Control is passed back to the calling statement (the statement that called the method).

__3___ Control is passed to the first statement in the method body.

__1___ The arguments are evaluated left to right.

Question 4. (4 points) We've made the point several times that it is important that programs be well-written so that human readers can understand them.

(a) Why is this important? (Give at least one specific reason)

Human readers need to be able to understand the code to fix bugs, add features, change the operation of the program in response to new requirements, etc.

(b) Give two specific examples of things that a program author should do to help make code easier to understand. (Think about things that make it easier or harder for someone to understand how a program works and what it is doing.)

Here are several possibilities

- **Pick meaningful names for things**
- **Indent appropriately to indicate program structure**
- **Include comments to specify methods and their parameters**
- **Avoid tricky or obscure code**
- **Be consistent – do the same thing the same way everywhere it appears**
- **etc. …**

Question 5. (7 points) Here is a simple class with three methods.

```
public class Questionable {

   public void doThis( ) {

      int n = 42;                                  ___1___

      doSomethingElse(n);                          ___6___

      int m = n + 1;                               ___7___
   }

   public void doThat( ) {
      String fini = "almost done";                 ___3___
   }

   public void doSomethingElse(int n) {

      int one = 1;                                 ___2___

      doThat( );                                   ___4___

      int zero = one - one;                        ___5___
   }
}
```

Now, suppose that we create an instance (object) of class `Questionable` and call method `doThis()` for that object:

```
Questionable query = new Questionable();
query.doThis();
```

The question is, in what order do the statements in these methods ***finish*** execution when `doThis` is called? Fill in the blanks to the right of the statements to show the order in which the statements finish (1, 2, 3, …). If a method is not executed when `doThis` is called, leave the spaces for that method blank.

[The question asked for the order in which the statements finished, in order to be consistent with the lecture slide on control flow. If your answer gave the order in which the statements start (1, 2, 7, 5, 3, 4, 6), you received almost full credit.]

Question 6. (3 points) Normally, a name can only be created once. For example, the following code would generate a compiler error because the name x is declared twice:

```
Public class Tiny {

   public void zip() {
      int x = 12;     // fine
      …
      int x = 15;     // error – name already exists
   }
}
```

But it's possible for two different methods to use the same local name. For example,

```
public class Tim {

   public void one () {
      int n = 12;
      …
   }

   public void two() {
      int n = 142;
      …
   }
}
```

How is this possible? Why doesn't the second definition of `n` generate a "name already exists" error?

Each method defines a new scope. Names declared inside a method's scope are local to that method and are not visible in other methods.

[For full credit, your answer had to have more explanation than just saying "scope".]

Question 7. (20 points) This longer question asks you to implement methods to draw a spaceship (UFO).

[Question details omitted to save space]

(a) (6 points) Write a method named `drawUFOWindow` to draw one of the triangular windows. This method should have 4 integer parameters: the x and y coordinates for the center of the UFO window, and the width and height of the triangle.

```
/** Draw one triangular UFO window
* @param x horizontal center of window
* @param y vertical center of window
* @param width width of window
* @param height height of window
*/
public void drawUFOWindow(int x, int y, int width, int height) {
    theWindow.add(new Triangle(x, y-height/2,
                               x+width/2, y+height/2,
                               x-width/2, y+height/2,
                               Color.white, true));
}
```

(b) (14 points) Write a method named `drawUFO` to draw the complete UFO. This method should have 7 integer parameters: The x and y coordinates of the center of the ship's body; the height and width of the ship body, the radius of the dome that sits on top of the ship, and the height and width of the triangular windows. (Continue on the back side of the page if necessary.) For full credit, you **must** call method `drawUFOWindow` from part (a) to draw the three windows. The ship can be any solid color you like.

```
/** Draw a UFO
* @param x horizontal center of the UFO
* @param y vertical center of the UFO
* @param domeRadius radius of the dome on top of the UFO
* @param bodyHeight height of the main part of the UFO
* @param shipWidth  width of the ship
* @param winWidth width of each triangular window
* @param winHeight height of each triangular window
*/
public void drawUFO(int x, int y, int domeRadius, int bodyHeight,
                    int shipWidth, int winWidth, int winHeight) {
    // dome
    theWindow.add(new Oval(x - domeRadius,
                           y - (bodyHeight/2 + domeRadius),
                           2*domeRadius, 2*domeRadius,
                           Color.blue, false));

    // body
    theWindow.add(new Oval(x-shipWidth/2, y-bodyHeight/2,
                           shipWidth, bodyHeight,
                           Color.cyan, true));

    // triangular windows
    drawUFOWindow(x-shipWidth/4, y, winWidth, winHeight);
    drawUFOWindow(x, y, winWidth, winHeight);
    drawUFOWindow(x+shipWidth/4, y, winWidth, winHeight);
}
```