R2.1   
Explain the difference between an object and an object reference.

An "object" is an entity in a program that is manipulated by calling methods. An "object reference" is the memory location of that an object, which is referred to by a variable.   
The expression new Rectangle(10, 20, 30, 40); is an object.   
The expression Rectangle r = new Rectangle(10, 20, 30, 40); creates the object reference *r*.

R2.2   
Explain the difference between an object and an object variable.

An "object variable" refers to an object reference, which is the location in memory of an object.

R2.3   
Explain the difference between an object and a class.

An object is an entity in a program that is manipulated, and objects belong to "classes," which specify the methods that can be applied to objects in that class. The .length() method, for example, can be applied to any object that is in the String class.

R2.4   
Give the Java code for constructing an *object* of class Rectangle, and for declaring an *object variable* of class Rectangle.

Rectangle box = new Rectangle();

R2.5   
Explain the difference between the = symbol in Java and in mathematics.

In Java, the = symbol is used to assign a value to an identifier (or variable)--it's sometimes called the *assignment operator*. In mathematics, the = symbol is an assertion of equality, that the terms on one side of an equation have the same value as the terms on the other side of the equation.

R2.6   
Uninitialized variables can be a serious problem. Should you always initialize every int or double variable with zero? Explain the advantages and disadvantages of such a strategy.

It doesn't make sense to initialize a variable to zero if that's not its initial value. That just means that we'll have to change the variable's value to some other value later on in the program. The advantage to initializing to zero is that we won't have uninitialized variables in our programs, but the disadvantage is that we're just going to have to reset that value later on, which makes reading the program harder to understand.

R2.7   
Give Java code to construct the following objects. Create objects, not object variables. a. A rectangle with center (100, 100) and all side lengths equal to 50

The object is new Rectangle(75, 75, 50, 50)   
The object variable would be:   
Rectangle myRect = new Rectangle(75, 75, 50, 50);

b. A string â€œHello, Dave!â€‌   
The object is "Hello, Dave!"

R2.8.   
Repeat Exercise R2.7, but now define object variables that are initialized with the required objects.

a. Rectangle myRect = new Rectangle(75, 75, 50 50);

b. String greeting = "Hello, World";

R2.9.   
Find the errors in the following statements:   
a.   
Rectangle r = (5, 10, 15, 20);   
This appears to be an attempt to re-initialize *r*, or the programmer is trying to initialize *r* but has left off some the *new* constructor. It should probably be written:   
Rectangle r = new Rectangle(5, 10, 15, 20);

b.   
double width = Rectangle(5, 10, 15, 20).getWidth();   
The rectangle has to be created with a *new* constructor before we can get its width. This should be two separate statements:

Rectangle r = new Rectangle(5, 10, 15, 20);

double width = r.getWidth();

c.   
Rectangle r;   
r.translate(15, 25);

The rectangle hasn't been constructed yet. At a minimum we need to use *new* to create *r*:

Rectangle r = new Rectangle(); // sets up a default rectangle

r.translate(15, 25);

d.   
r = new Rectangle();   
r.translate("far, far away!");

The data type of *r* is missing in the constructor statement, and the translation method needs to have numbers for *x* and *y* values as parameters, not the string "far, far away!"

R2.10.   
Name two accessor methods and two mutator methods of the Rectangle class.

Two accessor methods would be getWidth() and getHeight(); also getX() and getY().   
Two mutator methods would be translate() and setLocation(). (Look in the API documentation for info on these.)

R2.11   
Look into the API documentation of the Rectangle class and locate the method

void add(int newx, int newy)

Read through the method documentation. Then determine the result of the following statements:

Rectangle box = new Rectangle(5, 10, 20, 30);

box.add(0, 0);

If you are not sure, write a small test program or use BlueJ.

Looking at the API doscumentation at [http://java.sun.com/javase/6/docs/api/index.html], it appears that the add() method extends the boundaries of the box. Writing a small test program in BlueJ allows us to test this idea.

R2.12   
Find an overloaded method of the String class.

A method of a class is overloaded when it can be used with different parameter types. For the String class, by looking at the API documentation one can see that contentEquals() method can be used with either a character sequence or a string buffer (we haven't learned about those yet), and the valueOf() method can be used with a wide variety of parameters. There are other overloaded methods for the String class as well.

R2.13   
Find an overloaded method of the Rectangle class.

For the Rectangle class, the method add() is overloaded. It can be used with x-y coordinates, with a Point, or with another Rectangle.

R2.14   
What is the difference between a console application and a graphical application?

A console application runs in the console, ie. the Terminal window. It's strictly text-based, with plain text arranged in straight lines. A graphical application, however, displays drawings inside a window frame on the screen.

R2.15   
Who calls the paintComponent method of a component? When does the call to the paintComponent method occur?

The window calls the paintComponent method. The paintComponent method is called whenever the component needs to be repainted--when the window is resized, for example.

R2.16   
Why does the parameter of the paintComponent method have type Graphics and not Graphics2D?

Graphics is a primitive data type used by paintComponent. It is simpler than the Graphics2D class, which provides object-oriented painting features.

R2.17   
What is the purpose of the graphics context?

The purpose of the graphics context is to allow for the display of non-text-based information: pixels, lines, shapes, etc.

R2.18   
Why are separate viewer and component classes used for graphical programs?

The viewer class cannot be worked on directly. Components have to be created separately, and then added to the viewer window using the frame.add() method.

R2.19   
How do you specify a text color?

You specify the text color the same way that you specify the color of anything else. First use the setColor() method to establish the color:

g2.setColor(Color.RED);

... and then draw the string onto the component, where the text will have the indicated color:

g2.drawString("Hello, I'm red!",5, 100);