Sample Exam 3 Name: _____

Answer each question in the space provided. Point values are listed next to each question number. You may use your textbook and class notes while taking this exam.

1. What is the difference between an instance method and a static method?

2. Explain the meaning of the word “this” in a Java program, and give two examples of how you might use it in a class:

3. Given the class definition fragment below, **write a default constructor** (one that takes no arguments) **that calls the existing constructor**:

```
public class Mystery
{
      private int onion;
      private String borscht;
      public Mystery(int o, String b)
      {
            onion = o;
            borscht = b;
      }
…
```

Your code here:

4. Use the class below to determine if instructions a through i are valid; **if so, write "valid" in the space next to the instruction and if not, explain what is wrong <u>or</u> correct the error**. Each subsequent instruction can assume that previous instructions were valid:

```
public class Albatross
{
      private int wings;
      private String bellow;
      public Albatross()
      {
            wings=2;
            bellow= "awk";
      }
      public void setFire(int match)
      {
            wings = (int)(Math.random()*match);
      }
      public String getAway()
      {
            return bellow;
      }
}
```

Code (in main method, different class)	Error/Correction
a) Albatross a = new Albatross(-4, "whoop");	
b) System.out.println(a);	
c) Albatross.getAway();	
d) int x = a.setFire(100);	
e) a.getOut("of here");	
f) String shout = a.getAway();	
g) a.setFire(a.getAway());	
h) a.setFire(wings);	
i) a.getAway("eh?");	

5. Write a main method that fully tests the code in the class provided. Your method should call each of the methods in the Code class at least once, and allow the user to run the tests as many times as they want to with their own data:

Code class:

```
public class Code
{
    private int ascii,
                unicode;
    private char encoded;

    public Code(char c)
    {
        this.setChar(c);
    }
    public Code()
    {
        this.setChar((char)((int)
            (Math.random() * 1000
             % 255)));
    }
    public char getEncoded()
    {
        return encoded;
    }
    public int getAscii()
    {
        return ascii;
    }
    public int getUnicode()
    {
        return unicode;
    }
    public void setChar(char c)
    {
        encoded = c;
        unicode = (int)c;
        ascii = unicode % 255;
    }
    public void setChar(int i)
    {
        unicode = i;
        encoded = (char)i;
        ascii = unicode % 255;
    }
    public String toString()
    {
        return encoded + " "
             + ascii + " " + unicode;
    }
}
```

Your main method:

6. Write a Java class that models a Pixel (one of the many colored dots that makes up the display on a computer screen). Instance variables must include:
    - red, green and blue: int values (in the range 0..255) that represent the saturation of each of the colors represented by a pixel
    - size: an int value between 1 and 50, representing the size of a pixel; we can think of a pixel as a size x size solid square in the designated color

    Instance methods (these must enforce the integer limits described above):

    - 3 constructors:
        - default constructor creates a 1 x 1 white pixel (NOTE: white is full color saturation; all 3 color values are 255);
        - constructor with 4 int arguments, representing red, green, blue and size;
        - constructor with an int argument representing size, and default color of white
    - Mutator methods to set size, set individual color values, and set all 3 color values
    - Accessor methods to return values for colors & size
    - a toString method that returns a String with this format: **red:xxx, green:xxx, blue:xxx, size:xx** (where the x's are the actual int values)
    - an equals method that returns true if the calling object and a Pixel argument are the same size, false otherwise

You do ***not*** need to write a main method – just the Pixel class itself, with the elements described above.

Scratch paper: use as needed