# CS 273 Laboratory 10: Inheritance

This lab gives you experience with Java class inheritance. It has a total of 100 possible points.

## Preliminaries

Open the project and run FaceFrame. It should display a window with 7 cartoonish faces that are identical. Each is labeled (e.g., "SIMPSON", "HOMER", "MARGE") to indicate whose face it is supposed to be. Right now, all the faces are the generic FACE.

If you do not see the entirety of all 7 faces, open FaceFrame.java and locate the Y\_COORD\_ROW constants. As an example, if you make the value for Y\_COORD\_ROW\_2 smaller, it will move the third row of faces up. Adjust the constants as needed to be able to see all of the faces.

Examine the files in the project:

* FaceFrame.java - an application that creates the faces and displays them
* Face.java - the definition of a “Face” object. Study all the methods in this class. Note especially that there are methods (e.g., pixelX, pixelY, distX, distY) for computing coordinates relative to positions within the current object's head (e.g., 20% of the way across the face). These will be handy when drawing facial features.
* Polygon2.txt - The file is the documentation for a Polygon2 class, which is a custom Polygon class, which, in addition to normal Polygon functionality, allows the following:
  + polygons that approximate a crescent-moon to be created
  + rotation
  + horizontal and vertical scaling

Study this documentation so that you can use the Polygon2 class effectively.

A Face object draws a face within a given bounding box. The face is drawn with:

* a pink head
* blue eyes
* a red rectangular mouth
* a red triangular nose
* an upside-down half-circle (Polygon2 object) of yellow hair
* pink ears

There are seven face classes:

* Face
* SimpsonFace
* HomerFace
* MargeFace
* HillFace
* HankFace
* BobbyFace

The reason that they all display the same face is that all are declared as subclasses of Face, and the starter code does not define each type of face to have any additional (or different) behavior.

**Important**: You should not modify Face.java or FaceFrame.java during this lab (unless you attempt the extra credit).

## Laboratory

### Part 1: Update SimpsonFace

If you are not familiar with the long-running cartoon The Simpsons, please find a photo online of this classic cartoon family. Your first task will be to create a face (SimpsonFace) that contains many features that are the same (or close enough for our purposes) for all members of this family.

Edit SimpsonFace.java so that a SimpsonFace object acts like a Face object except for the following differences:

1. A SimpsonFace is narrower and longer
2. The head color is yellow
3. The mouth is a black smile

You should update the SimpsonFace class using the following steps:

1. Update the constructor so that the width is multiplied by 0.75 and the height is multiplied by 1.25 before being passed into super. You will have to typecast to comply with the expected parameter types.
2. Override the mouthColor method from the Face class by defining a protected method called mouthColor in the SimpsonFace class that takes no parameters and returns a black Color object.
3. Create a protected method called createMouth that takes no parameters and returns a Polygon2 object in the shape of a smile. (See the documentation for the Polygon2 class with regard to the constructor and methods to scale and rotate the Polygon2 object.) The object returned should fit in the face. Recall that the method call pixelX(20) returns the x-coordinate location that is 20% from the left edge of the face’s bounding box.
4. Override the drawMouth method from the Face class by defining a protected method called drawMouth in the SimpsonFace class that takes a Graphics object g as a parameter and returns nothing. The drawMouth method in the SimpsonFace class should set the color of the graphics pen to this.mouthColor(). Then the method should create a Polygon2 variable and assign it to the result of calling createMouth(). Finally, the method should draw the mouth to the Graphics object g by calling fillPolygon with the mouth polygon as the argument.

**checkpoint 1 (30 points): Show your lab instructor or assistant that the two happy faces now look as described above.**

### Part 2: Complete SimpsonFace

Update SimpsonFace further so that it looks as follows:

1. Set the head to the official “Simpson yellow”
   1. Override the headColor method from the Face class by defining a protected method called headColor in the SimpsonFace class that takes no parameters and returns a Color object with RGB values of (254, 212, 29).
2. There is no hair
   1. Do this by overriding the drawHair method.
3. The eye whites are much larger and close together
   1. Override the drawEyeWhites method from the Face class. Set the color to white and draw two large circles close together for the eyes. Approximate the eyes of the actual Simpsons characters to the best of your ability.
4. The eye color is black and in the center of the large white parts
   1. Override the eyeColor method.
   2. Override the drawEyeCenters method. Use the eyeColor method to set the color.
5. The nose is black and a different shape
   1. Override the noseColor method.
   2. Override the drawNose method. Use the noseColor method to set the color. I recommend a rotated crescent shape for the nose, but you are welcome to use something that may be a better approximation of the actual Simpson characters.

**checkpoint 2 (30 points): Show your lab instructor or assistant that the SimpsonFace now looks as described above.**

### Part 3: Create Homer’s Face

Edit HomerFace.java so that a HomerFace object is drawn just like a SimpsonFace object except that:

* There are at least a couple of black hairs on Homer’s head
* There is a brown patch around Homer’s smile. The official color is RGB (209, 178, 113).

Use **subclassing** and **inheritance** whenever possible. For instance, Homer’s smile is unchanged from the SimpsonFace smile. Perhaps you could leverage inheritance there.

**checkpoint 3 (20 points): Show your lab instructor or assistant that Homer’s face now looks as described above.**

### Part 4: Create Marge’s Face

Edit MargeFace.java so that it is drawn just like a SimpsonFace object, except that:

* Marge has a tall stack of blue hair on the top of her head and along the side of her head. The hair must have ridges, so it must be more elaborate than a single shape. See if you can find the official color of her hair.

Again, use **subclassing** and **inheritance** whenever possible.

**checkpoint 4 (20 points): Show your lab instructor or assistant that Marge’s face**

### EXTRA CREDIT (optional)

### EC1: Marge’s Red Pearl Necklace (5 points)

### Edit MargeFace.java to include Marge’s red pearl necklace. While you could choose to override a method like drawMouth to include the necklace, a better way to do it would be to create a drawNecklace method in the MargeFace class. Then, in the MargeFace class, override the method from the Face class that draws all the parts of the face together and include your drawNecklace method.

**checkpoint EC1 (5 points): Show your lab instructor/assistant Marge’s red pearl necklace.**

### EC2: New Family (10 points)

### There are three faces (Hill, Hank, Bobby) on the right half of the frame that we didn’t touch in this lab. (They were originally intended to be characters from King of the Hill.) Draw a family of your choosing.

### HillFace must have at least 3 features different from Face.

### HankFace must have at least 1 feature different from HillFace.

### BobbyFace must have at least 1 feature different from HillFace and HankFace.

### Update the class names and labels to represent the characters you drew.

**checkpoint EC2 (10 points): Show your lab instructor/assistant your customized family.**