CSE 8A Lecture 11

• Reading for next class: Videos! Check the website
• PSA6 is out. Go to discussion section tomorrow!
• Today:
  – Review: Variables/scope
  – Live coding!

Did you go to discussion yesterday?
A. Yes
B. No
“Green screen” special effects

• Actors or other objects are filmed in front of a screen with a very specific shade of green

• Later, a computer program checks the color of each pixel of the movie
  – Pixels matching that specific shade of green are replaced with pixels from another image/movie
  – Pixels not matching that specific shade of green are left alone
“Green screen” special effects

- Actors can wear green and be invisible in the final movie
  - The two men on the left are moving a puppet
  - The man on the right, if he doesn’t put his green shirt back on, will appear to be a floating torso
1. What does the following print

```java
int x = 10;
int y = 20;
if (x < 10 || y > 10) {
    System.out.println("Hello");
} else {
    System.out.println("Goodbye");
}
```

A. Hello
B. Goodbye
C. Nothing is printed
2. What does the following print?

```java
int x = 10;
if(x<9) {
    x = x + 3;
}
if(x>10) {
    x = x + 1;
}
System.out.println(x);
```

A. 10  
B. 11  
C. 13  
D. 14
3. What does the java.lang.ArrayIndexOutOfBoundsException tell you?

A. Nothing of importance
B. That your array is too large
C. That you forgot a semi-colon somewhere.
D. That you tried to access an element that doesn't exist in the array.
4. What gets printed in the following:

```java
int x = 5;
if(x < 6)
{
    x = x+2;
}
else if(x>6)
{
    x = x + 1;
}
System.out.println(x);
```

A. 5  
B. 6  
C. 7  
D. 8
Now quick review of a few concepts
(They will ALWAYS be important)

• Parameters
• Return values
• Variable scope
• Good commenting style
It’s nice to have code that is “user controllable”…

We have been hard-coding constants (40, 3, 100, for example) a lot, but we can write more flexible code using PARAMETERS

This lets us write code to do things like “cropping and pasting into a blank canvas”, but letting the user specify what part of the source picture to crop, and where to place it in the canvas.
Underline the values you would change into parameters and write a new method header

```java
public void copyKatiesXXX(Picture sourcePic,
    int sxMin, int sxMax, int syMin, int syMax)
{
    Pixel sPixel, tPixel = null;
    for (int sX = 40, tX = 100; sX < 110; sX++, tX++)
    {
        for (int sY = 350, tY = 100; sY < 400; sY++, tY++)
        {
            sPixel = sourcePic.getPixel(sX,sY);
            tPixel = this.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor();
        }
    }
}
```
Using parameters

In Picture.java...

```java
public void copyRegionTo (Picture target, int xSource, int ySource, int xTarget, int yTarget )
{
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = target.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();

Write the code to copy the square at position (10, 50) in fish to the blank canvas at position (30, 30) (vote on next slide)```
Using parameters

In Picture.java...

```java
public void copyRegionTo (Picture target, int xSource, int ySource, int xTarget, int yTarget )
{
    // Body omitted to save space
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();

A. fish.copyRegionTo(blank, 10, 50, 30, 30)
B. fish.copyRegionTo(blank)
C. blank.copyRegionTo(fish, 10, 50, 30, 30)
D. blank.copyRegionTo(fish, 30, 30, 10, 50)
E. None of these
```

Write the code to copy the square at position (10, 50) in fish to the blank canvas at position (30, 30)
In Picture.java...

```java
public void copyRegionTo (Picture target, int xSource, int ySource, int xTarget, int yTarget )
{
    // Body omitted to save space
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();
fish.copyRegionTo(blank, 10, 50, 30, 30);
```

Variables **only exist** in the region they are defined. I.e. variables in main cannot be accessed in copyRegion and vice versa. The region where a variable exists is called its scope.
In Picture.java...

```java
public void copyRegionTo (Picture target, int xSource, int ySource, int xTarget, int yTarget )
{
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX, sY);
            tPixel = target.getPixel(tX, tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();
fish.copyRegionTo(blank, 10, 50, 30, 30);
```

Main’s variables

- fish
- blank

BEFORE the call
public void copyRegionTo (Picture target, int xSource, int ySource, int xTarget, int yTarget)
{
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = target.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
}

In Picture.java...

In main...

Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();
fish.copyRegionTo(blank, 10, 50, 30, 30);
Parameters and scope

In Picture.java...

```java
public void copyRegionTo (Picture target, int xSource, int ySource, int xTarget, int yTarget )
{
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = target.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture blank = new Picture();
fish.copyRegionTo(blank, 10, 50, 30, 30);
```
In Picture.java...

```java
public Picture copyRegionToNew(int xSource, int ySource, int xTarget, int yTarget )
{
    Picture newCanvas = new Picture();
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = newCanvas.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture newCanvas = fish.copyRegionToNew(10, 30, 50, 50);
newCanvas.show();
```

What error will the following code produce?

A. This code will not compile
B. The line “Picture newCanvas = fish.copyRegionToNew…” in main will cause an error
C. The line newCanvas.show() will cause an error
In Picture.java...

```java
public Picture copyRegionToNew(int xSource, int ySource, int xTarget, int yTarget )
{
    Picture newCanvas = new Picture();
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = newCanvas.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
    return newCanvas;
}
```

In main...

```java
Picture fish = new Picture( "fish.jpg" );
Picture newCanvas = fish.copyRegionToNew(10, 30, 50, 50);
newCanvas.show();
```
Parameters and scope

In Picture.java...
public Picture copyRegionToNew(int xSource, int ySource, int xTarget, int yTarget )
{
    Picture newCanvas = new Picture();
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = newCanvas.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
    return newCanvas;
}

In main...
Picture fish = new Picture( "fish.jpg" );
Picture myNewCanvas = fish.copyRegionToNew(10, 30, 50, 50);
myNewCanvas.show();
Some comments on comments

/* A method to copy a 100x100 region of the calling object’s
* image to a blank canvas.
* xSource, ySource: the upper left corner of the
*     region to be copied.
* xTarget, yTarget: the upper left corner where the region
*     will appear in the new canvas.
* returns a new canvas with the region copied into it. */

public Picture copyRegionToNew(int xSource,
         int ySource, int xTarget, int yTarget )
{
    Picture newCanvas = new Picture(); // create a new picture
    Pixel sPixel, tPixel = null;
    for (int sX = xSource, tX = xTarget; sX < 100+xSource; sX++, tX++)
    {
        for (int sY = ySource, tY = yTarget; sY < 100+ySource; sY++, tY++)
        {
            sPixel = this.getPixel(sX,sY);
            tPixel = newCanvas.getPixel(tX,tY);
            tPixel.setColor(sPixel.getColor());
        }
    }
    return newCanvas;
}
Challenge: Create a checkerboard

Write a method that will copy a pattern of 20x20 squares from a source image to a new Picture, and return the new picture.
Challenge: Create a checkerboard

/** copy the calling object image to the target picture, in a
 * Checkerboard pattern of 20x20 pixels.
 * input: int height, int width – the height and width of the new
 * Picture to create
 * returns: a new Picture with the specified dimensions
 */
public Picture copyCheckerboard( int width, int height) {

Where to start? What are the main tasks we need to accomplish?
(Discuss with your group)
Challenge: Create a checkerboard

/**
copy the calling object image to the target picture, in a
* Checkerboard pattern of 20x20 pixels.
* input: int height, int width - the height and width of the new
* Picture to create
* returns: a new Picture with the specified dimensions
*/
public Picture copyCheckerboard(int width, int height)
{

Where to start? What are the main tasks we need to accomplish?
1. Create a new picture
2. We need a loop that copies some of the pixels
3. Return the new picture
Modular development

/** copy the calling object image to the target picture, in a
 * Checkerboard pattern of 20x20 pixels.
 * input: int height, int width – the height and width of the new
 * Picture to create
 * returns: a new Picture with the specified dimensions
 */

public Picture copyCheckerboard( int width, int height) {
    Picture toReturn = new Picture( width, height );

    return toReturn;
}

1. Create a new picture
2. We need a loop that copies some of the pixels
3. Return the new picture

Tasks don’t need to be implemented in order!
Modular development

/** copy the calling object image to the target picture, in a
* Checkerboard pattern of 20x20 pixels.
* input: int height, int width – the height and width of the new
* Picture to create
* returns: a new Picture with the specified dimensions
*/
public Picture copyCheckerboard( int width, int height)
{
    Picture toReturn = new Picture( width, height);

    return toReturn;
}
/** copy the calling object image to the target picture, in a
Checkerboard pattern of 20x20 pixels.
* input: int height, int width – the height and width of the new
* Picture to create
* returns: a new Picture with the specified dimensions
*/
public Picture copyCheckerboard( int width, int height)
{
    Picture toReturn = new Picture( width, height );
    for ( int x = 0; x < width ; x++ ) {
        for (int y = 0; y < height ; y++ ) {

            Pixel source = this.getPixel( x, y );
            Pixel target = toReturn.getPixel( x, y );
            target.setColor( source.getColor() );
        }
    }
    return toReturn;
}
/** copy the calling object image to the target picture, in a 
* Checkerboard pattern of 20x20 pixels.
* input: int height, int width – the height and width of the new 
* Picture to create
* returns: a new Picture with the specified dimensions 
*/

public Picture copyCheckerboard( int width, int height )
{
    Picture toReturn = new Picture( width, height );
    for ( int x = 0; x < width ; x++ ) {
        for (int y = 0; y < height ; y++ ) {

            Pixel source = this.getPixel( x, y );
            Pixel target = toReturn.getPixel( x, y );
            target.setColor( source.getColor() );
        }
    }

    return toReturn;
}
public Picture copyCheckerboard( int width, int height) {
    Picture toReturn = new Picture( width, height );
    for ( int x = 0; x < width && x < this.getWidth(); x++ ) {
        for (int y = 0; y < height && y < this.getHeight(); y++ ) {

            if ( ____________________________________________ ) {
                Pixel source = this.getPixel( x, y );
                Pixel target = toReturn.getPixel( x, y );
                target.setColor( source.getColor() );
            }
        }
    }
    return toReturn;
}
How to draw the boxes?

```java
public Picture copyCheckerboard( int width, int height )
{
    Picture toReturn = new Picture( width, height );
    for ( int x = 0; x < width && x < this.getWidth(); x++ ) {
        for (int y = 0; y < height && y < this.getHeight(); y++ ) {

            if ( x / 20 == 2 && y / 20 == 2 ) {
                Pixel source = this.getPixel( x, y );
                Pixel target = toReturn.getPixel( x, y );
                target.setColor( source.getColor() );
            }
        }
    }

    return toReturn;
}
```

What does the above code draw? (Discuss)
Can you figure out how to modify the if-statement so that it draws the checkerboard?
How to draw the boxes?

public Picture copyCheckerboard( int width, int height) {
    Picture toReturn = new Picture( width, height );
    for ( int x = 0; x < width && x < this.getWidth(); x++ ) {
        for (int y = 0; y < height && y < this.getHeight(); y++ ) {
            if ( (x / 20) % 2 == 0 && (y / 20) % 2 == 0 ) {
                Pixel source = this.getPixel( x, y );
                Pixel target = toReturn.getPixel( x, y );
                target.setColor( source.getColor() );
            }
        }
    }
    return toReturn;
}
TODO

• Reading for next class: Videos!
• Work on PSA6