• **IN TERM EXAM 1 THIS THURSDAY**
  – Will cover everything through today’s class (especially last week readings, and in-class)
  – Similar to PI, reading questions. End of chapter questions good to study.
  – Not too much on methods yet. Focus on variables, assignment, objects, references, and Turtle commands.
  – Review in Discussion tomorrow. Also watch Piazza…

• PSA2 due today midnight; next assignment posted tonight/tomorrow
  – To find a partner for PSA3… post on the Piazza forum

• Interviews for PSA2 deadline Friday noon.

• Slip day: you can use it (no need to ask)

• Turnin: turn in as many times as you want, but not after deadline

• All PSAs worth the same. Interview always 20% of PSA grade.

• Seating charts: Express constraints on Piazza
Lab 2 this week

• Practice with random numbers and loops (but also get more experience with variable declaration and assignments)

• Seating charts! Check your email and arrive early to “find” your seat in the lab.
Tutor Lab Hours

• Covers many hours a day, in B230 (website has the schedule)

• Procedure:

1. Arrive in Lab, work (with partner).
2. When you have a question, look on the board for the tutor queue.
3. Add your name and location
4. Tutor will find you

CLICKERS OUT!
1. Consider the following code that gets the width of a picture and stores it in a variable \( w \). Assume `pict` already holds a reference to a Picture object.

\[
w = \text{pict.getWidth}();
\]

What type of data does the method `getWidth()` return (what should the type of \( w \) be)?

A. Pixel  
B. int  
C. double  
D. String
2. How many times does the body of this loop execute?

```java
int index = 10;
while (index >= 0)
{
    System.out.println("CSE8A ROCKS!");
    index = index - 1;
}
```

A. 0  
B. 9  
C. 10  
D. 11

READING QUIZ - NO TALKING
3. You want to create a for loop to modify all of the pixels in pixelArray. What would go in the blank area to do this?

```java
Pixel[] pixelArray = this.getPixels();
Pixel pixel = null;
for (int index = 0; ______________ index++;)
{
    pixel = pixelArray[index];
    pixel.setColor(Color.black);
}
```

A. `index < pixelArray.length`
B. `index < pixelArray.length - 1`
C. `index < pixelArray`

READING QUIZ - NO TALKING
4. A pixel is colored Color(255,0,100). Which value represents the Green value of the pixel?

A. 255
B. 0
C. 100
Today in CSE 8A

- Random?
- Thinking *loopily* for a while
- More memory models

Pictures start Thursday
Chapter 2 review!

• What is stored in the variable `name` after the following line of code is executed?

```java
String name = "Bob";
```

A. "Bob"
B. Bob
C. A reference to where "Bob" is stored in memory
D. `new String("Bob")`

**Hint:** Are Strings objects or primitives?
Primitives vs. Objects: Review

• What does the following code print?

```cpp
int remzisAge = 19;
int rominasAge = 25;
remzisAge = rominasAge;
rominasAge = 20;
print("Remzi is " + remzisAge + " and Romina is " + rominasAge)
```

A. Remzi is 19 and Romina is 25
B. Remzi is 20 and Romina is 20
C. Remzi is 25 and Romina is 20
D. Remzi is 25 and Romina is 25
E. Remzi is 19 and Romina is 20

FILL IN THE VARIABLE BOXES!!
EVERYONE, GET OUT A PEN AND PAPER!!
CS Concept: Primitives vs. Objects

**Primitives**

```java
int remzisAge = 19;
int rominasAge = 25;
remzisAge = rominasAge;
```

**Objects**

```java
World world1 = new World(200,100);
Turtle maria = new Turtle(25, 25, world1);
Turtle jose = new Turtle(100, 50, world1);
jose = maria;
```
For Reference: The anatomy of a method

**Method header**

```
public void drawSquare(int size)
{
    this.turnLeft();
    this.forward(size);
    this.turnLeft();
    this.forward(size);
    this.turnLeft();
    this.forward(size);
    this.turnLeft();
    this.forward(size);
}
```

- **Return type**: `public void`
- **name**: `drawSquare`
- **Parameter list**: `(int size)`

**Method body**

- `this` refers to the “calling object” (i.e. the object that the method is eventually called with)
Writing methods == adding power (to the Turtle, in this case)

In file Turtle.java

```java
public class Turtle
{
    // lots of methods not shown

    public void drawSquare(int size)
    {
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
    }
}
```

In file TurtleTester.java

```java
public class TurtleTester
{
    public static void main(String[] args)
    {
        World w = new World();
        Turtle jane = new Turtle(w);
        Turtle sean = new Turtle(w);
        jane.drawSquare(100);
        sean.drawSquare(50);
    }
}
```

(world not shown)
Passing parameters and calling methods: IMPLICIT ASSIGNMENT!

```java
public class Turtle {
    public void drawSquare(int size) {
        // size = 100
        // this = jane
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
    }
}

public class TurtleTester {
    public static void main(String[] args) {
        World w = new World();
        Turtle jane = new Turtle(w);
        Turtle sean = new Turtle(w);
        jane.drawSquare(100);
        sean.drawSquare(50);
    }
}
```
Passing parameters and calling methods: IMPLICIT ASSIGNMENT!

In file Turtle.java

```java
public class Turtle {
    public void drawSquare(int size) {
        // size = 100
        // this = jane
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
    }
}
```

In file TurtleTester.java

```java
public class TurtleTester {
    public static void main(String[] args) {
        World w = new World();
        Turtle jane = new Turtle(w);
        Turtle sean = new Turtle(w);
        jane.drawSquare(100);
        sean.drawSquare(50);
    }
}
```

When `jane.drawSquare(100)` is called, these two lines happen implicitly

// size = 100
// this = jane
Passing parameters and calling methods: IMPLICIT ASSIGNMENT!

In file Turtle.java

```java
public class Turtle {
    public void drawSquare(int size) {
        // size = 100
        // this = jane
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
    }
}
```

In file TurtleTester.java

```java
public class TurtleTester {
    public static void main(String[] args) {
        World w = new World();
        Turtle jane = new Turtle(w);
        Turtle sean = new Turtle(w);
        jane.drawSquare(100);
        sean.drawSquare(50);
    }
}
```

In file Turtle.java

```java
public class Turtle {
    public void drawSquare(int size) {
        // size = 100
        // this = jane
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
    }
}
```

In file TurtleTester.java

```java
public class TurtleTester {
    public static void main(String[] args) {
        World w = new World();
        Turtle jane = new Turtle(w);
        Turtle sean = new Turtle(w);
        jane.drawSquare(100);
        sean.drawSquare(50);
    }
}
```
public class Turtle {
    public void drawSquare(int size) {
        // size = 50
        // this = sean
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
        this.turnLeft();
        this.forward(size);
    }
}

public class TurtleTester {
    public static void main(String[] args) {
        World w = new World();
        Turtle jane = new Turtle(w);
        Turtle sean = new Turtle(w);
        jane.drawSquare(100);
        sean.drawSquare(50);
    }
}
Random Numbers

HINT: Random is a class that works as a generator of random numbers.

• What do you think the following code prints?

```java
import java.util.Random;

// declare a generator for random numbers
Random generator = new Random();
// generate a random number
int value1 = generator.nextInt(10);
double value2 = generator.nextDouble();
System.out.println(value1); // (1)
System.out.println(value2); // (2)
```

A. This will not compile
B. 0 and 0.0
C. 1 and 2.0
D. 10 and 0.0
E. You cannot predict in advance
Random Numbers

• What do you think the following code prints?

```java
import java.util.Random;

// declare a generator for random numbers
Random generator = new Random();
// generate an int between 0 and 10, not including 10
int value1 = generator.nextInt(10);
// generate a double between 0.0 and 1.0, not including 1.0
double value2 = generator.nextDouble();
System.out.println(value1); // one possible value
5
System.out.println(value2); // one possible value
0.42
```

Needed for Lab and PSA3, but we’ll give you what you need, so don’t worry too much. (Not on Exam 1)
CS Concept: while loops

How many times is each section of code executed?

```
World world1 = new World(200, 100);
Turtle maria = new Turtle(25, 25, world1); }

int numberOfIterations = 5;
int index = 0;

while (index < numberOfIterations) }
{
    maria.forward(10);
    index = index + 1;
}
```
while loops and memory models

World world1 = new World(200, 100);
Turtle maria = new Turtle(25, 25, world1);

int numberOfIterations = 5;
int index = 0;

while (index < numberOfIterations)
{
    maria.forward(10);
    index = index + 1;
}
What does this code do?*

A. Creates 4 Turtles and moves them forward 12 units
B. Creates 1 Turtle and moves it forward 52 units
C. Creates 4 Turtles and moves them forward 52 units.
D. Creates 1 Turtle and moves it forward 12 units.

* This is something you WILL be asked to do on an exam/quiz (though not on Thursday). You should also be able to draw the memory model so do it now!
World world1 = new World(200, 100);
Turtle maria = new Turtle(25, 25, world1);
int numberOfIterations = 4;
int index = 0;
value = 10;
// this would do the same as the while loop:
  maria.forward(value);
  value = value+2;
  index = index + 1;

maria.forward(value);
value = value+2;
index = index + 1;

maria.forward(value);
value = value+2;
index = index + 1;

maria.forward(value);
value = value+2;
index = index + 1;

maria.forward(value);
value = value+2;
index = index + 1;
Summary: Rules of Memory Model Drawing

• There are two kinds of variables in Java memory:
  – Primitive type variables, and class type object reference variables:
    • primitive type variable “boxes” with actual values in them
    • Class/object variable “boxes” have an arrow in them pointing to an object, or are null
  – Objects themselves: class/object type ACTUAL DATA is stored here

• When you see a declaration (like `type varName;`):
  – Write down the name next to a box
  – Put value (primitive type) or pointing arrow (class type) when assigned

• When you see instantiation (creation) of an actual object (including arrays)
  – Draw larger box (or multi-boxes if array) and fill in values

• Execution of other kinds of statements
  – If an assignment is made, change the value in the box on the left side of the assignment
  – If you need to evaluate an expression, use the values currently in the boxes
Summary of Exam 1 topics

- Turtles and calling Turtle methods (WE WILL GIVE YOU A “CHEAT SHEET”)
- Sequential execution
- Tracing code
- References and objects
- Primitive types
- Working with variables
- Box and arrow diagrams/memory models
- Assignment statements
- Filenames
- Dr. Java
- Compiling and running code

PRACTICE WITH ALL CLICKER QUESTIONS YOU’VE SEEN SO FAR
TODO

• Finish PSA2
• Study for Thursday exam
• No reading for Thursday