• Reading for next class: 11.4-11.5

• Today’s goals:
  – More practice with designing classes
  – Tracing code and creating memory models

• PSA 9 (classes) due next Monday (3/11)
  – Individual (no partner)

• PSA8 due tonight (interview by Thursday)
A Point class

```java
public class Point
{
    private int x;
    private int y;

    public Point(int x_in, int y_in)
    {
        this.x = x_in;
        this.y = y_in;
    }

    public static void main( String[] args )
    {
        Point r = new Point(10, 20);
        Point p = new Point(3, r.x);
        Point q = p;
    }
}
```

Draw a picture of the memory model at the end of main (vote on next slide)
A. 

B. 

C. 

D. None of these 

E. I don’t know
```java
public class Point
{
    private int x;
    private int y;

    public Point(int x_in, int y_in)
    {
        this.x = x_in;
        this.y = y_in;
    }

    public static void main(String[] args)
    {
        Point r = new Point(10, 20);
        Point p = new Point(3, r.x);
        Point q = p;
        r.x = 1;
        q.y = r.x + p.x;
    }
}
```

What are the values of r, p, and q when this code completes?

<table>
<thead>
<tr>
<th></th>
<th>r</th>
<th>p</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>(1, 20)</td>
<td>(3, 1)</td>
<td>(3, 4)</td>
</tr>
<tr>
<td>B.</td>
<td>(1, 20)</td>
<td>(1, 1)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>C.</td>
<td>(10, 20)</td>
<td>(3, 13)</td>
<td>(3, 13)</td>
</tr>
<tr>
<td>D.</td>
<td>(1, 20)</td>
<td>(1, 13)</td>
<td>(1, 13)</td>
</tr>
<tr>
<td>E.</td>
<td>None of these</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Point of Java: objects and classes
How many errors are there in this code (and what are they)

A. 2
B. 3
C. 4
D. 5
E. >=6

```java
public class Species {
    private String name;

    public static void main(String[] args) {
        double[] population;
        double growthRate;
    }

    public Species() {
        String name = "No Name Yet";
        double[] population = {0,0,0,0,0,0,0,0};
        growthRate = 33.3;
    }
}
```
public class Species
{
    private String name;
    
    public static void main(String[] args)
    {
        private int[] population;
        private double growthRate;

        Instance variables should be private

        public Species()
        {
            String name = "No Name Yet";
            int[] population = {0,0,0,0,0,0,0,0};
            growthRate = 33.3;
        }
    }
}

All instance variables and methods have to go inside the class {}.

All instance variables have to go outside of method definitions.

Instance variables should be private.

Refer to instance variables in a constructor; do not redefine them.
Visibility of Instance Variables

• Class design rule of thumb: *make all instance variables* **private**
  
  – “**private**” means: visible only inside this class
  
  – So a **private** instance variable or instance method cannot be seen from outside the class
  
  – Making an instance variable **private** prevents incorrectly setting its value by malicious or careless users of the class
public class Species
{

///////// fields ///////////
private String name;
private int[] population;
private double growthRate;

///////// constructors ///////////
public Species()
{
    name        = "No Name Yet";
    population  = {0,0,0,0,0,0,0};
    growthRate  = 33.3;
}

///////// methods ///////////
}
Getter and Setter methods

• Q: Instance variables correspond to properties of an object… if they are private and hidden inside, how can they interact with other objects?

• A: Define public instance methods which give controlled, safe access to the private instance variables
  
  – If the method can change an instance variable, it is a “mutator” or “modifier” or “setter” method
  
  – If it only returns the value of an instance variable, it is an “accessor” or “getter” method
Which of following would you select for “getter” method signatures for Species class?

- `public void getName();`
- `public void getPopulation();`
- `public void getGrowthRate();`
- `public String getName();`
- `public int[] getPopulation();`
- `public double getGrowthRate();`
Which of the following would you select for “setter” method declarations for the `Species` class?

```
public void setName();
public void setPopulation();
public void setGrowthRate();
```

```
public String setName();
public int[] setPopulation();
public double setGrowthRate();
```

```
public void setName(String newName);
public void setPopulation(int[] newPop);
public void setGrowthRateRate(double newGrowthRate);
```

```
public void setName(String newName);
public boolean setPopulation(int[] newPop);
public void setGrowthRateRate(double newGrowthRate);
```
Return type for Setters

- A getter method should have a non-void return type

- A setter can be designed in several ways:
  - **void**: just change the values of the instance variable(s), don’t return anything
  - **boolean**: return true if the setting was successful and false if not (for example if setting would be ‘illegal’)
  - The type of the value that is being changed: return the previous value
Overloading:

Which are legal overloads?

A. 1

B. 2

C. 3

D. 1 and 3

E. 1 and 2

```java
public Species()
public Species(String newName);

public boolean setGrowthRate(double gr)
public void setGrowthRate(double gr)

public void setPopulation(int northAmerica,
                          int southAmerica,
                          int europe,
                          int asia,
                          int africa,
                          int australia,
                          int antarctica)

public void setPopulation(int[] a)
```
1. Declaration

2. Instantiation

3. Initialization

```java
double [] foo;
foo = new double[5];
for(int i = 0 ; i < foo.length ; i++)
{ foo[i] = -11.5; }
```
public Species( String newName, int[] newPop, double newGR )
{
    name       = newName;
    population = new int[newPop.length];

    for( int i=0 ; i< this.population.length ; i++ )
        population[i] = newPop[i];

    growthRate = newGR;
}
TODO

• Reading for next class: 11.4-11.5
• Submitted PSA8 (if you haven’t done it already)
• Start PSA9
  – Do this one individually (no team programming)