CSE 8A Lecture 19

• Reading for next class: Videos! (to help with PSA 10)

• Today’s goals:
  – More practice with designing classes
  – Tracing code and creating memory models
  – Overview of PSA10

• REMINDERS – important instructor announcements this week on Piazza, please read carefully:
  – NO LAB this week
  – Schedule for last week of classes (all the same, only section D00 will be in Pepper Hall 109)
  – CONFIRM exam changes requests
Exam 4

• Stats:
  – Mean: 8.68; Median: 9.5!
  – 114 perfect scores (10 out of 10)
  – Nice job!

• If you scored below 55% you are in danger of failing the course. Come talk to your instructor.
Problem 1

```java
public void gobblegobble() {
    SoundSample[] original = this.getSamples();
    for (int index=0; index<(original.length - 1); index++)
    {
        original[index].setValue(original[index].getValue() +
                                 original[index+1].getValue());
    }
}
```

What about the last element?
A. 0
B. 30
C. 35
D. Error! Out of range

original (BEFORE)

| 5 | 20 | 0 | 15 | 10 | 5 | 25 | 0 | 30 |

original (AFTER)

| 25 | 20 | 15 | 25 | 15 | 30 | 25 | 30 | ?? |
Problem 1

```java
public void gobblegobble() {
    SoundSample[] original = this.getSamples();
    for (int index=0; index<original.length - 1; index++) {
        original[index].setValue(original[index].getValue() +
                                  original[index+1].getValue());
    }
}
```

original (BEFORE)

```
| 5 | 20 | 0 | 15 | 10 | 5 | 25 | 0 | 30 |
```

original (AFTER)

```
| 25 | 20 | 15 | 25 | 15 | 30 | 25 | 30 | ?? |
```
Problem 2. What was wrong?

```java
public void increasePitch(Sound target) {
    int newPlace = 0;

    for (int index = 0; index < this.getLength(); index = index + 2) {
        target.setSampleValueAt(newPlace, this.getSampleValueAt(index));
    }
}
```

What variable do you think is *useless* here?

A. index
B. newPlace
C. target
Problem 3

Example

result = original.increaseVolumeFirstHalfByFactor(1.5);

<table>
<thead>
<tr>
<th>original</th>
<th>5</th>
<th>20</th>
<th>0</th>
<th>10</th>
<th>10</th>
<th>5</th>
<th>25</th>
<th>0</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>result:</td>
<td>7</td>
<td>30</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

public Sound increaseVolumeFirstHalfByFactor(double factor) {
    Sound target = new Sound(this);
    SoundSample[] targetSamples = target.getSamples();
    SoundSample[] original = this.getSamples();
    for (int index = 0; index < original.length; index++) {
        if (index < original.length/2) {
            targetSamples[index].setValue((int)(original[index].getValue()*factor));
        } else {
            targetSamples[index].setValue(0);
        }
    }
    return target;
}

Can I avoid using this if-else statement?

A. Yes
B. No
C. No, never
Problem 3

Example
result = original.increaseVolumeFirstHalfByFactor(1.5);

<table>
<thead>
<tr>
<th>original</th>
<th>5</th>
<th>20</th>
<th>0</th>
<th>10</th>
<th>10</th>
<th>5</th>
<th>25</th>
<th>0</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>result:</td>
<td>7</td>
<td>30</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

public Sound increaseVolumeFirstHalfByFactor(double factor) {
    Sound target = new Sound(this);
    SoundSample[] targetSamples = target.getSamples();
    SoundSample[] original = this.getSamples();
    for (int index = 0; index < original.length; index++) {
        if (index < original.length/2) {
            targetSamples[index].setValue((int)(original[index].getValue() * factor));
        }
        else {
            targetSamples[index].setValue(0);
        }
    }
    return target;
}

Do I need two loops if I don’t use the if-else statement?
   A. Yes, that’s the only way
   B. No, never
   C. No if I use other Sound constructor
public class Species{

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

    ////////////// constructors /++++++++++++
    public Species(String name, int[] pop, double gr)
    {
        this.name = name;
        population = new int[pop.length];
        for (int i=0; i< this.population.length;i++)
            population[i] = pop[i];
        growthRate = gr;
    }

    ////////////// methods /++++++++++++++++++
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;
}
// ...In main
int[] pops = {20, 30, 1, 4, 2, 6, 11};
Species s = new Species( "Oompa", pops, 1.2 );
The Species class, continued

///// methods ///////////////

public void setPopulation(int pop, int index) {
    population[index] = pop;
}

public int getPopulation(int index) {
    return population[index];
}
A redesign of the Species class

• This idea that the population array is just 7 entries, one per location is a bit “obscure”.
  – What are the names of the locations? Which entry is for North America? Which for Europe?

• Another, better approach: “parallel arrays”

• Declare and create two arrays of the same length
  – One for location names: String[] location;
  – One for population numbers: int[] population;
  – And write code so that for every index I, population[I] is the population in the location with name location[I]
Write a constructor for the new Species class

```java
public Species(String name, int[] pop, String[] location, double gr) {
    this.name = name;
    population = new int[pop.length];
    for (int i=0; i< population.length; i++)
        population[i] = pop[i];

    <<INSERT CODE HERE>>
    growthRate = gr;
}
```

A) `location = location;`

B) `location = new String[location.length];
    for (int i=0; i < location.length; i++)
        location[i] = location[i];`

C) `this.location = location;`

D) `location = new String[location.length];
    for (int i=0; i < location.length; i++)
        this.location[i] = location[i];`

E. None of the Above
Write a constructor for the new Species class

```java
public Species(String name, int[] pop, String[] location, double gr) {
    this.name = name;
    population = new int[pop.length];
    for (int i=0; i< population.length; i++)
        population[i] = pop[i];

    this.location = new String[location.length];
    for (int i=0; i < location.length; i++)
        this.location[i] = location[i];

    growthRate = gr;
}
```

Still has some “bad software design issues…”
public Species(String name, int[] pop, String[] location, double gr) {
    name = newName;
    growthRate = gr;
    if (pop.length != location.length) {
        System.out.println(“Error constructing Species. “ + “Population array and location array must be same length.”);
        population = null; location = null;
        return;
    }
    population = new int[pop.length];
    this.location =
        new String[location.length];
    for (int i=0; i < location.length; i++) {
        this.location[i] = location[i];
        this.population[i] = pop[i];
    }
}
A possible setter method

```java
public boolean setPopulation(int pop, String loc) {
    if (pop < 0)  return false;
    for (int i=0; i<loc.length; i++)
    {
        if (location[i].equals(loc))
        {
            population[i] = pop;
            return true;
        }
    }
    return false;
}
```
PSA – 10: Turtle path planning

- Basically consists on finding the way from the start to the goal locations
PSA – 10: Turtle path planning

• We will use a Map (grid of cells)
  – Way of accessing this grid is similar to access pixels in a Picture: top-left is (0,0); bottom-right is (width-1, height-1);

• We actually have two matrices
  – Occupancy
  – Cost
PSA – 10: Turtle path planning

• How to get to the red Cell?
PSA – 10: Turtle path planning

• How to get to the red Cell?
PSA – 10: Turtle path planning

- How to get to the red Cell?

A) COST?
B) DISTANCE?
PSA – 10: Turtle path planning

• Important ideas:
  – **COST**: how many “steps” (we count one cell up, down, left, right as one “step” left)
  – **PATH**: array of cells to follow (Cell[] path = new Cell[n])
  – **NEIGHBORS**: We consider 4 neighbors (up, down, left, right)
PSA – 10: Turtle path planning

• Basic idea:
  – We start in the “goal” cell: that cell has $cost = 0$.
  – Every “free” neighbor has cost+1
  – The neighbor of that neighbor has cost+1+1
  – ...
  – Propagate the wave until we find the cost from the start location.
PSA – 10: Turtle path planning

- How to get to the red Cell?
PSA – 10: Turtle path planning

• Steps:
  1. Load map from a .txt file (all 0s and 1s). Initialize occupancyMap (0: free; 1: occupied)
  2. Compute costMap (int[][] costMap) given startCell and endCell
     • This method is given
     • Once we run it, we already know the length of the path!

A) costMap.getWidth() * costMap.getHeight()
B) getCostValue (endCell) + 1
C) getCostValue (startCell) + 1
PSA – 10: Turtle path planning

• Steps:

1. Load map from a .txt file (all 0s and 1s). Initialize occupancyMap (0: free; 1: occupied)

2. Compute costMap (int[][] costMap) given startCell and endCell
   • This method is given
   • Once we run it, we already know the length of the path!

3. Find the path (fill the Cell[] path) looping on the values of costMap.
   • Start in startCell, until we reach the endCell

4. Make the turtle move from cell to cell to traverse all cells in the path in order.
Next week Tuesday class and Lab are focused on helping with PSA10.
  - But try to get started earlier! It’s challenging but you can certainly do it.

Start studying for the final!