Welcome to More Classes and Space Game Lab

Learning Outcomes

By the end of this lab:

- Developing an instantiable class
- Coding constructors, accessors, mutators
- Adding instance variables
- Programming for a continuous animation
- Game programming

Exercise @: Evaluation

1. Please fill out the TA Evaluation for any TAs with which you have had significant interaction.
2. Please fill out the Course and Instructor evaluation https://aefis.wisc.edu/. The instructors and CS Department will review these and have made significant changes in the past that have benefited you this semester.

Exercise A: Trace and Explain

Trace and Explain the following code without using Java Visualizer or an IDE. In Exercise B, you will get the opportunity to extend and play the game in an IDE.

Procedural programming vs Object-oriented programming

Most of this semester, we have introduced procedural programming emphasizing using class (static) methods and passing data between them using parameters. Now we are introducing object-oriented programming which emphasizes grouping related data and methods into objects/instances.

1. SpaceGameMain and UFO classes
    A. Find the following:
        a. class method, instance method, class variable, instance variable, constructor, accessor, mutator, constant
    B. Which class must be instantiated in order to call its methods?
    C. What information is necessary in order to instantiate a UFO?
    D. When a UFO is instantiated, which variables are allocated on the heap?
    E. What classes, other than SpaceGameMain and UFO, are required for this program? How do you know?
    F. Given a reference to an instance of UFO, how would you get the current location (x and y coordinates)?

//Note: In Exercise B, a more complete version of the code is available to be downloaded.

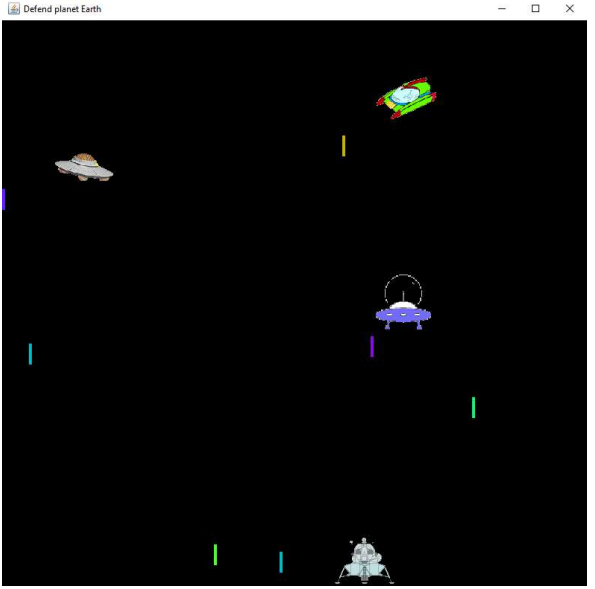
```
import java.util.Random;

public class SpaceGameMain {
    public static Random rng = new Random();
    public static void main(String[] args) {
```

```
        SpaceGame theGame = new SpaceGame(800,800);

        theGame.addUFO(UFO.SIMPLE_SAUCER);
        theGame.addUFO(UFO.SIMPLE_SAUCER);
        theGame.addUFO(UFO.SIMPLE_SAUCER);

        theGame.start();
    }
}

public class UFO {
     public static final int SIMPLE_SAUCER = 0;

     private int xPosition; //current x position of UFO's center
     private int yPosition; //current y position of UFO's center
     private int UFOType;   //type of UFO

     /**
      * Constructs a UFO object given its type and initial position
      *
      * @param startX initial x coordinate
      * @param startY initial y coordinate
      * @param UFOType int representing the type of UFO.
      */
     public UFO(int startX, int startY, int UFOType) {
          //TODO: TASK 1 - write your code here
     }
     /**
      * Get the current x coordinate of the center of this UFO.
      *
      * @return The current x coordinate of the center of this UFO.
      */
     public int getXPosition() {
          //TODO: TASK 1 - write your code
          //replace the return statement below
          return -1;
     }

     /**
      * Get the current y coordinate of the center of this UFO.
      *
      * @return The current y coordinate of the center of this UFO.
      */
     public int getYPosition() {
          //TODO: TASK 1 - write your code
          //replace the return statement below
          return -1;
     }
```

```
/**
 * Get an int represented the type for this UFO
 *
 * @return An int representing the type of this UFO
 */
public int getUFOType() {
      //TODO: TASK 1 - write your code
      //replace the return statement below
      return -1;
}

/**
 * Updates the position of the UFO for the next time it is redrawn.
 *
 * @param defender The Defender object. Ignore for now, but it may
 * be used later in the lab to determine movement of some UFO types.
 */
public void takeOneStep(Defender defender) {
      //TODO: TASK 2 - write your code here
}
```

Exercise B: Space Lab

Apply object-oriented programming skills to implement the UFOs in a Space Invaders like game with the following tutorial: http://pages.cs.wisc.edu/~cs200/labs/SpaceLab/SpaceLab.pdf This may take a while so keep track of how far you get and leave enough time to discuss the TA exercise.



Exercise TA: Demonstration and Discussion

This exercise is utilized to determine 3 points of the lab grade. We suggest reviewing these before you discuss these with your TA.

1. Exercise A: Trace and Explain
    a. Identify the various members of the classes and the answers for the questions.

2. Exercise B: Space Lab
    a. Show your TA how far you were able to get.

Exercise C: More Trace and Explain 

With your partner, trace and explain each of the following (also found in traceExplain2.txt). Check your understanding using Java Visualizer or Eclipse.

1. Find the following:
    a. variables: instance, class, parameter, local
    b. methods: instance method, class method, accessor, mutator
    c. no-arg constructor
    d. class visible outside the default package
    e. class visible only in the default package
    f. method visible only to other classes within the same package
    g. method visible to classes outside the same package
    h. post-increment operator

- What is the default value for toppings if no toppings are provided to the constructor?
- Will numPizzas be incremented when calling the no-argument constructor?

```
class Pizza {
    private String toppings;
    private static int numPizzas = 0;

    public Pizza() {
        this("cheese");
    }
    public Pizza(String toppings) {
        this.toppings = toppings;
        numPizzas++;
    }
    public String getToppings() {
        if (this.toppings == null)
            this.toppings = "no toppings";
        return this.toppings;
    }

    void setToppings(String newToppings) {
        if (!newToppings.contains("anchovies")) {
            this.toppings = newToppings;
        }
    }
    public String toString() {
        return this.toppings;
    }

    public static int numPizzas() {
        return numPizzas;
    }
```

```
}

public class MakePizza {
    public static void main(String[] args) {
        Pizza one = new Pizza();
        one.setToppings("bacon, pineapple");

        Pizza two = new Pizza("pepperoni");
        two.setToppings("sausage, anchovies");

        ArrayList<Pizza> pizzas = new ArrayList<>();
        pizzas.add(one);
        pizzas.add(two);
        System.out.println(pizzas);

        System.out.println( Pizza.numPizzas());
    }
}
```

2. Where is the bug? What are two ways to fix it?

```
public class Can {
    private boolean isOpen;
    public Can(boolean isOpen) {
        isOpen = isOpen;
    }
    public boolean isOpen() {
        return isOpen;
    }

    public static void main(String []args) {
        Can can = new Can( true);
        System.out.println( "can is open: " + can.isOpen());
    }
}
```

Additional Learning Materials

When you have mastered everything in this lab, *and in previous labs*, then you are welcome to learn from additional learning resources available on the web and beyond this course:

- https://codingbat.com/java
- https://www.khanacademy.org/computing/computer-programming
- https://techdevguide.withgoogle.com/
- https://code.org/
- http://programmingbydoing.com/

Note: Team Lab is focused on pair programming and open discussion and so it is not appropriate to work on individual programming assignments.

Faculty Associates contributed the Space Lab exercise.