Drawing classes and objects

Model 1: Memory Allocation

All programs require some amount of memory (RAM) to run. To keep things organized, memory is divided into segments.

Segment	Allocated	Freed	Example
data	when the program starts	when the program ends	`static int count = 0;`
stack	when methods are called	when methods return	`double area = 0.0;`
heap	when objects are created	when objects are destroyed	`new Color(0, 0, 0)`

Questions (30 min)

1. Review and discuss the attached source code for `Polygon`. For each variable declaration, indicate where its contents will be stored: in the data (D), on the stack (S), or in the heap (H).

Number of variable declarations in Polygon.java: _____

2. In the space below, draw a memory diagram to show the state of the `Polygon` program at the end of the `main` method. Make sure you put each box in the right segment.

Data *Heap*

Stack

3. Draw a UML diagram for the `Polygon` class. Refer to examples from last week's activity and/or the current programming assignment.

4. What information does the UML diagram convey? In contrast, what information does the memory diagram convey?

7. Explain what the following compiler error means.

```
public static void main(String[] args) {
    Polygon.sides = 4;
}
```

Error: non-static variable sides cannot be referenced from a static context

8. In your own words, explain what `static` means with respect to classes and objects.

Model 2: Memory Usage

When you declare a variable of a *primitive* type, Java immediately knows how much memory it will need to reserve for that variable.

```
char letter;   // 2 bytes (16-bit Unicode)
int count;     // 4 bytes (32-bit integer)
double score;  // 8 bytes (64-bit floating-point)
```

When you declare a variable of a class type, Java needs to allocate memory twice: once for the *reference*, and once for the *object*. In the latter case, Java may not know how much memory it will need to reserve until the program runs.

```
String name;           // 8 bytes for reference (on 64-bit machines)
name = in.nextLine();  // memory usage depends on what the user types
```

Questions (20 min)

9. To the right of each box on your memory diagram, write the number of bytes needed for that box. You may assume that each Color object requires 28 bytes and that Strings require 2 bytes per character stored in the String.

(It is impossible to be completely accurate with these numbers. Java objects require some memory overhead beyond the memory required for the instance variables. Exactly how much overhead will depend on the implementation details of the Java Virtual Machine.)

10. Determine how much memory the `Polygon` program will need (in bytes).

Data: _____ + Stack: _____ + Heap: _____ = Total: _____

11. In the space below, draw UML diagrams for `Location`, `Alien`, and `MIB` from Lab18.

12. On the back of this paper, draw a memory diagram for the `MIB` program, including how much memory each box requires.

Data: _____ + Stack: _____ + Heap: _____ = Total: _____

```
import java.awt.Color;

public class Polygon {

    public static final int MIN_SIDES = 3;
    public static final int MAX_SIDES = 100;

    private String name;
    private int sides;
    private Color color;

    public Polygon(String name, int sides, Color color) {
       this.name = name;
       this.sides = sides;
       this.color = color;
    }

    public String toString() {
       return this.name + " (" + this.sides + " sides)";
    }

    public static void main(String[] args) {
       Polygon p1 = new Polygon("triangle", 3, new Color(0, 0, 0));
       Polygon p2 = new Polygon("rectangle", 4, new Color(255, 0, 0);
       System.out.println(p1);
       System.out.println(p2);
    }

}
```

```
public class Location
{
    public static final Location JMU
        = new Location(38.435427, -78.872942);
    public static final Location ISAT
        = new Location(38.434663, -78.863732);

    private double latitude;
    private double longitude;

    /**
     * Explicit value constructor for this Location.
     *
     * @param latitude The latitude in degrees
     * @param longitude The longitude in degrees
     */
    public Location(double latitude, double longitude)
    {
        this.latitude = latitude;
        this.longitude = longitude;
    }

    /**
     * Are the two values within .000001 of each other?
     *
     * @return true if the two values are the same in
     *         the two objects and false otherwise
     */
    public boolean equals(Location other)
    {
        return (this.latitude - other.latitude <= .000001
             && this.longitude - other.longitude <= .000001);
    }

    /**
     * Returns the latitude and longitude for this Location.
     *
     * @return string representation of this Location
     */
    public String toString()
    {
        return String.format("%.6f/%.6f", latitude, longitude);
    }
}
```

```
public class Alien {

    private static int alienCount;
    private static int rogueCount;

    private String name;        // original name
    private String homePlanet;  // the home planet
    private Location location;  // where the alien is located
    private String alias;       // name that they go by on earth
    private boolean wanted;     // if the MIB is looking for them

    public Alien() {
    }

    public Alien(String name, String home, Location loc, String alias) {
    }

    public void capturedJMU() {
    }

    public void deport() {
    }

    public Location getCurrentLocation() {
    }

    public static int getNumberAliens() {
    }

    public static int getNumberWanted() {
    }

    public void goneRogue() {
    }

    public void move(Location newLoc) {
    }

    public String toString() {
    }
}
```

```
public class MIB {
    public static void main(String[] args) {
        Alien frank, bug, et, superman, steve, pete;

        frank = new Alien("Frank", "Saturn", Location.JMU, "Frank the Pug");
        bug = new Alien("Bug", "Romulus",
                        new Location(38.89778, -77.0132), "Edgar");

        System.out.println("Aliens now 1: " + Alien.getNumberAliens());
        System.out.println("Wanted now 1: " + Alien.getNumberWanted());

        frank.move(Location.ISAT);
        bug.goneRogue();

        System.out.println("Aliens now 2: " + Alien.getNumberAliens());
        System.out.println("Wanted now 2: " + Alien.getNumberWanted());

        et = new Alien(";opm**", "Home", new Location(38.8619, -77.0647), "ET");
        superman = new Alien("Kai-el", "Krypton",
                             new Location(40.7306, -73.9889), "Clark Kent");

        System.out.println("Aliens now 3: " + Alien.getNumberAliens());
        System.out.println("Wanted now 3: " + Alien.getNumberWanted());

        et.deport();
        superman.goneRogue();

        System.out.println("Aliens now 4: " + Alien.getNumberAliens());
        System.out.println("Wanted now 4: " + Alien.getNumberWanted());

        bug.deport();

        System.out.println("Aliens now 5: " + Alien.getNumberAliens());
        System.out.println("Wanted now 5: " + Alien.getNumberWanted());

        steve = new Alien("Beedle", "Jupiter", new Location(38.89778, -77.0132),
                          "Steven Spielberg");
        steve.capturedJMU();
        superman.capturedJMU();

        System.out.println("Aliens now 6: " + Alien.getNumberAliens());
        System.out.println("Wanted now 6: " + Alien.getNumberWanted());

        pete = steve;  // will we have new aliens now?
        steve.move(Location.ISAT);
        steve.goneRogue();

        System.out.println("Aliens now 7: " + Alien.getNumberAliens());
        System.out.println("Wanted now 7: " + Alien.getNumberWanted());

        System.out.println("\nThe final set of aliens: ");
        System.out.println(frank);
        System.out.println(bug);
        System.out.println(et);
        System.out.println(superman);
        System.out.println(steve);
        System.out.println(pete);
    }
}
```