CSM 61B Fall 2017 — Objects & Arrays

Mentoring 3: September 11, 2017

1 Switcheroo

The **Golden Rule of Equals** says:

> Given variables `b` and `a`, the assignment statement `b = a` copies all the bits from `a` into `b`.

Passing parameters obeys the same rule: copy the bits to the new scope.

1.1 What is wrong with this definition of `swap`? How can we fix it?

```java
class SimpleSwap {
    public static void swap(int a, int b) {
        int temp = b;
        b = a;
        a = temp;
    }
    public static void main(String[] args) {
        int x = 2, y = 5;
        System.out.println("x: " + x + ", y: " + y);
        swap(x, y);
        System.out.println("x: " + x + ", y: " + y);
    }
}
```

1.2 How is this implementation of `swap` different?

```
class Coordinate {
    int x, y;
    Coordinate(int x, int y) {
        this.x = x;
        this.y = y;
    }
}

class SwapObject {
    public static void swap(Coordinate p) {
        int temp = p.x;
        p.x = p.y;
        p.y = temp;
    }
    public static void main(String[] args) {
        Coordinate p = new Coordinate(2, 5);
        System.out.println("p.x: " + p.x + ", p.y: " + p.y);
        swap(p);
        System.out.println("p.x: " + p.x + ", p.y: " + p.y);
    }
}
```

2 Flatter Me

Arrays are ordered sequences of fixed length. Unlike Python lists, the `length` must be known when creating an array.

```
int[] a = new int[3];
```

It is possible to initialize and fill an array in a single expression.

```
int[] b = new int[]{1, 2, 3};
```

Java can infer the type of the array from its context, yielding this shorthand.

```
int[] c = {1, 2, 3};
```

Uninitialized values have a default value like `0`, **`false`**, or **`null`**.

```
String[] c = new String[1];
c[0] == null;
```

2.1 Implement `middle`, which takes in **`int`**`[]` and returns the middle element. If no element is in the exact middle, return the element to the left middle.

```
public static int middle(int[] data) {




}
```

2.2 Write a method `flatten` that takes in a two-dimensional array `data` and returns a one-dimensional array that contains all of the arrays in `data` concatenated together.

```
public static int[] flatten(int[][] data) {




}
```

3 Pony DeTails

Loosely speaking, Java obeys the following rules for **variable lookup**:

1. Look in the local scope. In general, curly braces `{}` define a scope.
2. Look in the instance and class.
3. Look in the superclass.

3.1 What would Java display?

```java
public class Pony {
    String name;
    int age;
    public Pony(String s, int a) {
        name = s;
        age = a;
    }
    public void getDeTails() {
        String name = "getInfo";
        System.out.println("My name is " + name);
        System.out.println("My age is " + age);
    }
    public static void main(String[] args) {
        Pony pony = new Pony("Jerry", 300);
        pony.getDeTails();
    }
}
```

4 Samehorse *Extra Practice Midterm Question*

4.1 What would Java display? Draw a box-and-pointer diagram to find out!

```java
public class Horse {
    Horse same;
    String jimmy;
    public Horse(String lee) {
        jimmy = lee;
    }
    public Horse same(Horse horse) {
        if (same != null) {
            Horse same = horse;
            same.same = horse;
            same = horse.same;
        }
        return same.same;
    }
    public static void main(String[] args) {
        Horse horse = new Horse("you've been");
        Horse cult = new Horse("horsed");
        cult.same = cult;
        cult = cult.same(horse);
        System.out.println(cult.jimmy);
        System.out.println(horse.jimmy);
    }
}
```