CSM 61B Spring 2018

Objects & Arrays

Mentoring 3: February 7, 2018

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?

```
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 Dogs Yay

3.1
```
class Dog {
    public void walk() {
        System.out.println("The dog is walking");
    }
}
class Beagle extends Dog {
    @Override
    public void walk() {
        System.out.println("The beagle is walking");
    }
}
```

What would Java display?

(a)
```
Dog fido1 = new Dog();
fido1.walk();
```

(b)
```
Beagle fido2 = new Beagle();
fido2.walk();
```

(c)
```
Beagle fido3 = new Dog();
fido3.walk();
```

(d)
```
Dog fido4 = new Beagle();
fido4.walk();
```

3.2 What would each call in `Poodle.main` print? If a line would cause an error, determine if it is a compile-error or runtime-error.

```
class Dog {
    void bark(Dog dog) {
        System.out.println("bark");
    }
}

class Poodle extends Dog {
    void bark(Dog dog) {
        System.out.println("woof");
    }
    void bark(Poodle poodle) {
        System.out.println("yap");
    }
    void play(Dog dog) {
        System.out.println("no");
    }
    void play(Poodle poodle) {
        System.out.println("bowwow");
    }
    public static void main(String[] args) {
        Dog dog = new Poodle();
        Poodle poodle = new Poodle();

        dog.play(dog)
        dog.play(poodle)
        poodle.play(dog)
        poodle.play(poodle)

        poodle.bark(dog)
        poodle.bark(poodle)
        dog.bark(dog)
        dog.bark(poodle)

    }
}
```

4 Pokemon *Extra Practice*

4.1 Identify the errors that occur when running the code to the right.

```
Pikachu p = new Pikachu();
Pokemon a = p;
p = a;
a.eat();
a = new Squirtle();
a.attack();
((Squirtle) a).attack();
Pokemon z = new Pikachu();
Squirtle s = (Squirtle) z;
((Pokemon) p).attack(z);
```

```
public class Pokemon {
    public int hp, power;
    public String cry;
    public String secret;
    public Pokemon() {
        hp = 50;
        cry = "Poke?";
    }
    public Pokemon(String c, int hp) {
        cry = c;
        this.hp = hp;
    }
    public void attack(Pokemon p) {
        p.hp -= power;
    }
    public void eat() {
        System.out.println("nom nom");
    }
}
public class Pikachu extends Pokemon {
    public Pikachu() {
        hp = 100;
    }
    public Pikachu(int hp) {
        super("Pika pika pikachu", hp);
    }
    public void attack(Pokemon p) {
        p.hp = 0;
    }
    public void eat() {
        System.out.println("nom Pika nom");
    }
}
public class Squirtle extends Pokemon {
    public void attack() {
        System.out.println("Water gun!!");
    }
}
```