LAB 2

Constructors, Mutators, Accessors, and Basic Parsing

Download your lab2.zip

babbage.cs.missouri.edu/~mremtf/cs3330/downloads/lab2.zip

Lab Comments

Please don’t assume this class is a blowoff, start on assignments early. Grades will be out this Friday evening.

Lab 2: Don’t forget to do Javadoc Commenting for all methods even the trivial mutator and accessors. Submitting an assignment late will result in a zero.

What’s a Constructor

Constructors have one purpose in OOP: to create an instance of a class

Can’t return anything!

Constructors are not truly methods.

Attributes

The fields inside of the class. Usually these are only used within the class instantiated. A few exception such as the array in java with the length attribute.

What’s a Mutator Method

Allows us to update and initialize an attribute stored in the class.

What’s an Accessor Method

Allows us to get an attribute of a class.

Why not access the attribute directly?

We want to implement **encapsulation** and **security** between classes. The mutator will do error checking and validation before setting the attribute. The accessor will create a copy of the attribute making sure we don’t corrupt the one stored in our class.

What it all looks like in Java

```
public class Person {
    String name;
    int age;
    // Constructor in Java
    public Person (String name, int age) {
        setName(name);
        setAge(age);
    }
    // Mutator in Java
    private setName(String name) {
        this.name = name;
    }
    private setAge(int age) {
        if (age > 0) {
            this.age  = age;
        }
        else {
            this.age = 10;
        }
    }
    // Accessors in Java
    public String getName() { return this.name}

    public int getAge () { return this.age}
} // End Of Person Class
```

Simple Parsing

Java makes parsing so easy that you’ll smile! No more using strtok, and pointers everywhere like C.

The String Object has an awesome method called split that makes decomposing strings into parts easy. String[] split(String regex)

Example:

```
String line = “Lab 1 sucked...”
String[] strArray = line.split(“ “);
System.out.println(strArray[0] + “ “ + strArray[1] + “ ” + strArray[2]);
```

Outputs: Lab 1 sucked...

Converting strings into primitives!

Each primitive has a library for itself, so for int types there is an Integer class, for double types there is a Double class and so on. Each class above contains the ability to parse a String object into it’s respective variable type.

Example:

```
String GhostBusters = “5552368”;
int number = Integer.parseInt(GhostBusters);
System.out.println(number);
```

Lab Challenge

Assume we have a Movie class in our project with the constructor, Movie(String name, int attendance) and in our code, we have Movie myMovie = null;

Given the input below, how would I go about creating and storing the input in the myMovie instance variable?

Input:

String line = “Blade Runner,20000”;

Lab Challenge Solution

```
String[] strArray = line.split(“,”);
myMoive = new Movie(strArray[0], Integer.parseInt(strArray[1]));
```

or

```
String[] strArray = line.split(“,”);
String name = strArray[0];
int attendance = Integer.parseInt(strArray[1]);
myMoive = new Movie(name, attendance);
```