Assignment A2

Get handout from course Piazza, the pinned note A2 FAQs.
Objective:
- Get practice with Java functions.
- Learn about and use JUnit testing
  Given to you before A1, but due after A1. Provide flexibility, allow you to get ahead and learn Java early.
Groups. You can do A2 with 1 other person. FORM YOUR GROUP EARLY! Use pinned Piazza Note @5 to search for partner!

Homework HW1

The two short videos and HW1 are working beyond our expectations.
Very few people understood what we wanted when we asked you to tell us how the assignment statement was executed at the end of the last lecture.

By Monday morning at 9:30, over 1/3 of the class had submitted HW1. A sampling of about 25 of them showed that the lesson had been learned and that you were by and large grateful for it.
Gries has been trying for years to get this message across; this is the first real success.

Java OO (Object Orientation)

Python and Matlab have objects and classes.
Strong-typing nature of Java changes how OO is done and how useful it is. Put aside your previous experience with OO (if any).
This lecture:

First: describe objects, demoing their creation and use.
Second: Show you a class definition and how it contains definitions of functions and procedures that appear in each object of the class.
Third: Talk about keyword null.
Fourth (if there is time). Show you a Java application, a class with a “static” procedure with a certain parameter.

Homework

1. Study material of this lecture.
2. Visit course website, click on Resources and then on Code Style Guidelines. Study
   3. Documentation
      3.1 Kinds of comments
      3.2 Don’t over-comment
      3.4 Method specifications
         3.4.1 Precondition and postcondition
3. Spend a few minutes perusing slides for lecture 3; bring them to lecture 3.
Java OO

References to course text and JavaSummary.pptx
- Objects: B.1 slide 10-16
- Calling methods: B.2-B.3 slide 18
- Class definition: B.5 slide 11
- public, private: B.5 slide 11, 12
- Indirect reference, aliasing: B.6 slide 17
- Parameter vs argument: B.12-B.14 slide 14
- Methods may have parameters
- Method calls may have arguments

Text mentions fields of an object. We cover these in next lecture

Java Summary 

Object is associated with a window on your computer monitor

Evaluation of new-expression creates an object

Evaluation of:
```
new javax.swing.JFrame()
```
creates an object and gives its name the name of the object

If evaluation creates this object, value of expression is
```
JFrame@25c7f37d
```

2 + 3 + 4

```
JFrame@25c7f37d
```

A class variable contains the name of an object

If variable h contains the name of an object, you can call methods of the object using dot-notation:
- Procedure calls: h.show();
- Function calls: h.getX();

```
x = y;
y = h;
```

```
h = new javax.swing.JFrame();
h.setTitle("this is a title");
h.setLocation(0, 0);
h = JFrame@25c7f37d
```

A class variable contains the name of an object

```
public class C {
    declarations of methods (in any order)
}
```

Class definition C goes in its own file named C.java

On your hard drive, have separate directory for each Java project you write; put all class definitions for program in that directory. You'll see this when we demo.
First class definition

/** An instance (object of the class) has (almost) no methods */
public class C {
}

Then, execution of
C k;  
C k = new C();  
creates object shown to right and stores its name in k

Class extends (is a subclass of) JFrame

/** An instance is a subclass of JFrame */
public class C extends javax.swing.JFrame {
    ...
}

C: subclass of JFrame
JFrame: superclass of C
C inherits all methods that are in a JFrame

Inside-out rule for finding declaration

/** An instance is a subclass of JFrame with a function area */
public class C extends javax.swing.JFrame {
    /** Return area of window */
    @ Spec, as a comment
    public int area() {
        return getWidth() * getHeight();
    }
    ...
}

You know it is a function because it has a return type

Class definition with a function definition

Class definition with a procedure definition

Inside-out rule for finding declaration

/** An instance is a subclass of JFrame with more methods */
public class C extends javax.swing.JFrame {
    public int area() {
        return getWidth() * getHeight();
    }
    public void setWtoH() {
        setSize(getHeight(), getHeight());
    }
    ...
}

Function area in each object.  getWidth() calls function
get dimension in the object which it represents.

Object has 2 partitions:
One for JFrame methods, one for C methods

Easy re-use of program part!
**Using an object of class `Date`**

```java
/** An instance is a `JFrame` with more methods */
public class C extends javax.swing.JFrame {
    ...
    /** Put the date and time in the title */
    public void setTitleToDate() {
        setTitle(new java.util.Date().toString());
    }
}
```

An object of class `java.util.Date` contains the date and time at which it was created. It has a function `toString()`, which yields the data as a String.

**About null**

```java
v1 C@16
v2 null
```

null denotes the absence of a name.

v2.getName() is a mistake! Program stops with a NullPointerException.

You can write assignments like: `v1 = null;`
and expressions like: `v1 == null`

**Hello World!**

```java
/** A simple program that prints Hello, world! */
public class myClass {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

args is an array of String elements.

We explain static next week.
Briefly: there is only one copy of procedure `main`, and it is not in any object.