Homework HW1

The answers you handed in at the end of lecture 1 showed mass confusion. Perhaps 80% of you weren’t sure what to write. This was not graded! It was only to help us and you assess the situation.

Doing HW1 will eliminate the confusion. Piazza note @34, (find a link to it in the pinned Piazza Recitation/Homework note.)

Evaluation, Execution, Syntax, Semantics.
Presenting an algorithm in English (2.5 minutes).
Executing the assignment statement (2.5 minutes).
Do HW1 and submit on the CMS

PPT slides, JavaHyperText.

CMS. Visit course webpage, click “Links”, then “CMS for 2110”.
Download ppt slides the evening before each lecture, have them available in class. Please don’t ask questions on the piazza about that material the day before the lecture.
Got a Java question? See first if it’s answered on JavaHyperText

Try Java out in https://tryjshell.org

On Piazza note @39, Preston Rozwood talked about JShell. Problem: You need Java 9 to use it. Maybe next semester we’ll switch to 9.

Eric Wang then suggested using https://tryjshell.org
Where you can type in Java snippets and have them executed/evaluated. It’s not as easy to use as DrJava, but it can help in some cases.
You don’t need Java version 9 to use it. It’s not using the Java on your computer.

Thank, Preston and Eric!

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, a blueprint for objects, and how it contains definitions of methods (functions and procedures) that appear in each object of the class.
Third: Talk about keyword null.
Fourth: Introduce Exceptions

Homework

1. Study material of this lecture.
2. Visit JavaHyperText, click on Code Style. 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 JavaHyperText entries

- Objects: object
- Calling methods: method call
- Class definition: class
  - public, private: public private method
- Parameter vs argument: parameter, argument
- Inside-out rule
- Methods may have parameters
- Method calls may have arguments

Fields of an object may be mentioned. We cover these in next lecture

Evaluation of new-expression creates an object

Evaluation of

\[
\text{new javax.swing.JFrame}() \\
\]

creates an object and gives as its value the name of the object

If evaluation creates this object, value of expression is

\[
\text{JFrame@25c7} \\
\]

If variable \( h \) contains the name of an object, you can call methods of the object using dot-notation:

- Procedure calls: \( h.\text{show}() \)
- Function calls: \( h.\text{getX()} \)

\[
x = y; \\
g = h; \\
\]

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: \( h.\text{show}() \)
- Function calls: \( h.\text{getX()} \)

Class definition: a blueprint for objects of the class

Class definition: Describes format of an object (instance) of the class.

- Description of what the class is for: */
- Access modifier
- public means \( C \) can be used anywhere
- 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

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

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

Class extends (is a subclass of) JFrame

```java
/** 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

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

Easy re-use of program part!

Class definition with a function definition

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

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

Inside-out rule for finding declaration

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

Call on procedure because it has void instead of return type

Class definition with a procedure definition

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

`v1.getName()` is a mistake! Program stops with a `NullPointerException`

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

Intro to Exceptions

```java
7 int x= 5;
8 System.out.println("x is now "+x);
9 assert x== 6;
```

When the assert statement is executed and `x` is not 6, an object of class `AssertionError` is created and "thrown". It contains info needed to print out a nice message.

```
java.lang.AssertionError
at A0.main(A0.java:9)
```

```java
14 public static void m() {
15    int y= 5/0;
16 }
```

When `5/0` is evaluated, an object of class `ArithmeticException` is created and "thrown". It contains info needed to print out a nice message.

```
java.lang.ArithmeticException
at A0.m(A0.java:15)
```

You will learn all about exceptions in next week’s recitation!

- `java.lang.AssertionError`
- `java.lang.ArithmeticException`
- `java.lang.NullPointerException`
- `java.lang.IllegalArgumentException`
- `java.io.IOException`