**CS100J 1 February 2007. Customizing a class & testing**

- Fields (variables in a folder), and getter & setter methods. Secs 1.4.1 (p. 45) & 3.1 (pp. 105-110 only)
- Constructors. Sec. 3.1.3 (p. 111–112)
- Testing methods. Appendix 1.2.4 (p. 486)

**Quiz 2 on Tuesday:**
- How do you evaluate a new expression (see slide 6)?
- What is the purpose of a constructor (see slide 5)?

**Quote for the day:**
*There is no reason anyone would want a computer in their home.*  

The company was a huge player in computer hardware and software in CS academia in the 1970’s. The PDP machines were well known. The VAX had unix on it, and C, and Lisp. It was the main computer in most CS departments of any stature. DEC was bought by COMPAQ in the late 1990’s.

---

**Field: a variable that is in each folder of a class.**
We generally make fields private instead of public, so that they cannot be referenced from methods that are outside the class.

```java
public class Chapter {
    private String title; // Title of the chapter
    private int number; // Number of the chapter
    private Chapter previous; // previous chapter (null if none)
}
```

**Quiz 2 on Tuesday:**
- How do you evaluate a new expression (see slide 6)?
- What is the purpose of a constructor (see slide 5)?

**Getter and setter methods**

```java
/** An instance describes a chapter of a book */
public class Chapter {
    private String title; // Title of chapter
    private int number; // No. of chapter
    private Chapter previous; // previous chapter (null if none)

    /** = the title of the chapter */
    public String getTitle() {
        return title;
    }

    /** Set the title of the chapter to t */
    public void setTitle(String t) {
        title = t;
    }

    Getter methods (functions) get or retrieve values from a folder.
    Setter methods (procedures) set or change fields of a folder
}
```

**Purpose of a constructor:**

*To initialize (some) fields of a newly created folder*

```java
/** An instance describes a chapter of a book */
public class Chapter {
    private String title; // Title of chapter
    private int number; // No. of chapter
    private Chapter previous; // previous chapter (null if none)

    /** Constructor: an instance with title t, chapter number i, and previous chapter p (null if none) */
    public Chapter(String t, int i, Chapter p) {
        title = t;
        number = i;
        previous = p;
    }
}
```

**New description of execution of a new-expression**

```java
new Chapter()  
```
creates a folder but doesn’t allow us to say what values should be in it.

We would like to be able to say:

```java
new Chapter("I am born", 1, null)
```
to set the title to "I am born", the chapter number to 1, and the previous chapter to null.

For this, we use a new kind of method, the constructor.

---

For this, we use a new kind of method, the constructor.

```java
new Chapter("I am born", 1, null)
```
to set the title to "I am born", the chapter number to 1, and the previous chapter to null.

For this, we use a new kind of method, the constructor.

```java
new Chapter(String t, int i, Chapter c)
```
sets the title to t, the chapter number to i, and the previous chapter to c.

The name of a constructor is the name of the class. Do not put a type or void here.

---

**New description of execution of a new-expression**

```java
new Chapter("I am born", 1, null)
```
creates a new folder of class Chapter, with fields initialized to default values (0 for int, for example) – of course, put the folder in the file drawer.

2. Execute the constructor call

```java
Chapter("I am born", 1, null)
```

3. Use the name of the new folder as the value of the new-expression.

Memorize this new definition! Today! Now!
**Testing — using JUnit**

**Bug**: Error in a program.

**Testing**: Process of analyzing, running program, looking for bugs.

**Test case**: A set of input values, together with the expected output.

**Debugging**: Process of finding a bug and removing it.

Get in the habit of writing test cases for a method from the specification of the method even before you write the method.

A feature called JUnit in DrJava helps us develop test cases and use them. You have to use this feature in assignment A1.

```java
public class ChapterTester extends TestCase {
    /** A test method. 
     * (Replace "X" with a name describing the test. You may write as many "testSomething" methods in this class as you wish, and each * one will be called when testing.) */
    public void testX() {
        // Code for testX
    }
    ...
}
```

A few other methods that can be used are listed on page 488.

1. c1= new Chapter("one", 1, null);
   Title should be: "one"; chap. no.: 1; previous: null.
2. c2= new Chapter("two", 2, c1);
   Title should be: "two"; chap. no.: 2; previous: c1.

We need a way to run these test cases, to see whether the fields are set correctly. We could use the interactions pane, but then repeating the test is time-consuming.

To create a testing framework: select menu File item new Junit test case... At prompt, put in class name ChapterTester. This creates a new class with that name. Save it in same directory as class Chapter. The class imports junit.framework.TestCase, which provides some methods for testing.

Every time you click button Test in DrJava, this method (and all other testX methods) will be called.