Lecture 4

Classes
Readings for This Lecture

- Section 1.4, 1.5 in text
- Section 3.1 in text
- Plive activities referenced in the text

Please look at lecture summaries online
- Handouts are short version
- Presentation is everything I do in class

I correct slides after class
- Fix errors in the slides
- Clarify confusing points

Always good to read my slides after class

2/2/12
First Assignment Posted Tomorrow

• Due Tuesday, February 14
  ▪ Submit earlier so we can start iterative feedback process
  ▪ Labs and one-on-ones (next slide) can help you

• Work alone or with one partner
  ▪ Partners “group themselves” on the CMS
  ▪ Only one person submits the files.
  ▪ Partners must do the work together, sit next to each other, with each taking turns “driving” (writing the code)

• Academic Integrity
  ▪ Never look at someone’s code or show yours to someone else
  ▪ Never possess someone else’s code (except your partner)
One-on-One Sessions

- **Starting Monday:** 1/2-hour one-on-one sessions
  - Bring computer and work with instructor, TA or consultant
  - Hands on exercise to covering Classes to see what you understand and give you help
  - Like assignment, but **not for help on assignment itself**
- **Limited availability:** we cannot get to everyone
  - Students with experience or confidence should hold back
- **Sign up online in CMS:** first come, first served
  - Choose assignment One-on-One
  - Pick a time that works for you; will add slots as possible
Extended Review From Last Time

- p.getName()
  - Has value “W. White”
  - **Function**; gives value

- p.pay(250.0);
  - Sets owes to 0
  - **Procedure**; it does something
Class versus Object

Anatomy of a declaration + assignment statement:

```
int x = 2;
Point3d p = new Point3d();
```
• You can declare a class variable w/o using new
  ▪ Example: Point3d var3;

• Value in variable is **null**
  ▪ **null**: Absence of a name

• var3.getX() gives error!
  ▪ There is no name in var3
  ▪ Does not know which Point3d to access
  ▪ NullPointerException

2/2/12

Classes
Class Definition

- Describes the format of a folder (instance, object) of the class.

```java
/**
 * Description of what the class is for
 */

public class <class-name> {
    declarations of fields and methods (in any order)
}
```

- The class and every method has a comment of the form

  ```java
  /** specification */
  ```

- This is a Javadoc comment (Part of Lab next week).
An instance is a worker in a certain organization.

```java
public class Worker {
    private String lname; // Last name ("" if none; never null)
    private int ssn; // Social security #: in 0..999999999
    private Worker boss; // Immediate boss (null if none)
}
```

Field: A Variable in each Folder of a Class

Declarations of fields

Invariants: Properties that are always true

Note the `private` and `public` keywords.
They are important but we will explain them later.

2/2/12
/** Yields: worker’s last name*/

public String getName() {
    return lname;
}

/** Set worker’s last name to n
 * Cannot be null; can be “” */

public void setName(String n) {
    lname= n;
}

/** Yields: last 4 SSN digits, as int *
• Try writing it yourself.
• Full code on website

2/2/12
/** Yields: worker’s last name*/

public String getName() {
    return lname;
}

/** Set worker’s last name to n 
 * Cannot be null; can be “” */

public void setName(String n) {
    lname = n;
}

/** Yields: last 4 SSN digits, as int *

• Try writing it yourself.
• Full code on website

Getter methods (functions) get or retrieve values from a folder.

Setter methods (procedures) set or change fields of a folder
How Methods Work

- **Example**: `var1.getX()`
  - Gets object (folder) name from the variable
  - Searches class (file drawer) for object (folder)
  - Executes commands inside the method on that object

- Methods apply to the **object** (folder), not the variable!
  - Execute `var2.setX(8.2);`
  - Makes `var3.getX() == 8.2`

Memorize This!
Write it down several times.

```
var1 @4e0a1
  var2 @13fc8
  var3 @13fc8
```

```
Point3d
x 2.2
y 5.4
z 6.7
```

```
Point3d
x 3.5
y -2.0
z 0.0
```
Initializing the Fields of an Object (Folder)

- Creating a new Worker is now a multi-step process:
  - `Worker w = new Worker();`
  - `w.setName(“White”);`
  - `...`
- We would like to be able to use something like
  
  ```java
  Worker w = new Worker(“White”, 1, null);
  ```
  - Create a new Worker, sets the last name to “White”, the SSN to 0000000001, and the boss to `null`.
  - Need a special kind of method: `the constructor`
Creating a new Worker is now a multi-step process:

- Worker w = new Worker();
- w.setName("White");
- ...

We would like to be able to use something like:

Worker w = new Worker("White", 1, null);

Create a new Worker, sets the last name to "White", the SSN to 0000000001, and the boss to null.

Need a special kind of method: the constructor

Invariants must always be true. Always.

Purpose of the Constructor

- Initialize the fields of a newly created object
- Make sure that the invariants are true

Memorize This!
Write it down several times.
Example Constructor

/**
 * Constructor: an instance with last
 * name n (can’t be null, can be “”),
 * SSN s (an int in 0..999999999), and
 * boss b (null if none)
 */

public Worker(String n, int s, Worker b) {
    lname = n;
    ssn = s;
    boss = b;
}

no void or type!
new Worker(“White”, 1, null)

• Create a new object (folder) of class Worker
  ▪ Initializes fields to default values
  ▪ e.g. 0 for int, null for String
• Put the folder in file drawer
• Execute the constructor call Worker(“White”, 1, null)
  ▪ Executes the (assignment) commands in constructor body
• Uses **the name** of the object as the final value of this expression
Quiz Next Week

• All about definitions; taken from these slides
  ▪ Everything that says “Memorize This!”
  ▪ Want English descriptions of the steps

• How do method calls work?
  ▪ Handout slide 7

• What is the purpose of the constructor?
  ▪ Handout slide 9

• How is new evaluated?
  ▪ Handout slide 11