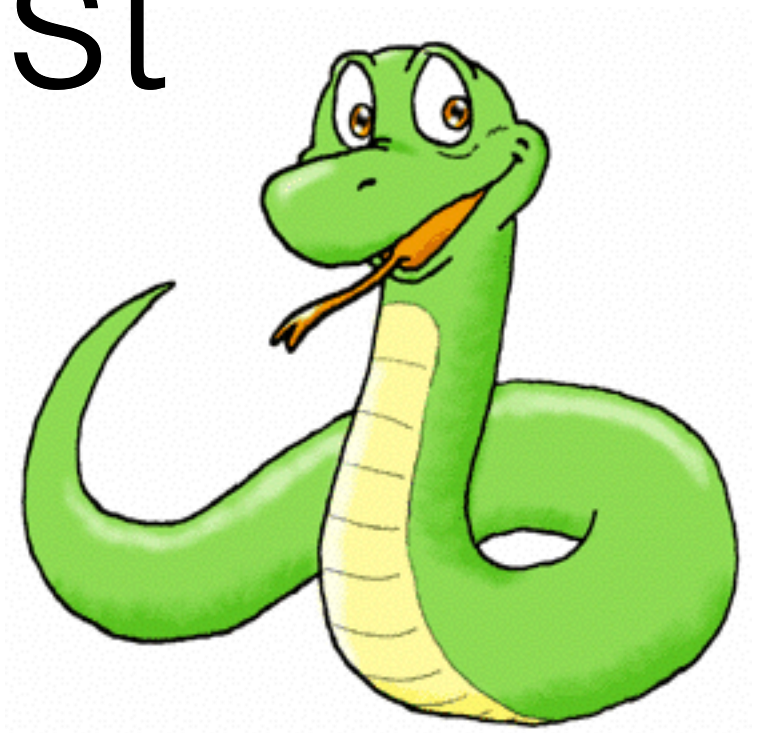
Introduction to Java

CSC212 Lecture 5
D. Thiebaut, Fall 2014

Back to our Python List



```
public class PythonList {
        int[] array = null;
        int maxdim;
        int end = 0;

        PythonList(int m) {
                maxdim = m;
                end = 0;
                array = new int[maxdim];
        }

        PythonList() {
                this(100);
        }
```

```java
public void append(int n) {
        if (end <= array.length - 1) {
                array[end++] = n;
                return;
        }
        int[] temp = new int[maxdim * 2];
        for (int i = 0; i < array.length; i++)
                temp[i] = array[i];
        array = temp;
        array[end++] = n;
        maxdim = maxdim * 2;
}

public int length() {
        return end;
}
```

```
        public int at(int index) {

                if (index >= 0 && index < array.length)
                        return array[index];
                // this is bad... we'll fix it later...
                else
                        return -1;
        }

        public static void main(String[] args) {
                // --- Test List ---
                PythonList L = new PythonList(5);

                for (int i = 0; i < 11; i++)
                        L.append(i);

                for (int i = 0; i < L.length(); i++)
                        System.out.println( L.at(i) );
        }
}
```

More Cementing

Understanding what is

private

versus what is

public

Private versus *Public*

A BAD EXAMPLE

```
public class Vehicle {

    public String brand;
    public int noWheels;      // should be private...
    public String type;       // should be private...

    Vehicle( String b, int n ) {
        brand = b;
        noWheels = n;
        type = getType( n );
    }

    private String getType( int n ) {
        if ( n==1 ) return "unicycle";
        if ( n==2 ) return "bicycle";
        if ( n==3 ) return "tricycle";
        if ( n==4 ) return "car";
        return "unknown";
    }

    public void display( ) {
        System.out.println( String.format( "%s (%d wheels) %s", brand, noWheels, type ) );
    }

    public static void main(String[] args) {
        Vehicle v1 = new Vehicle( "Shwin", 2 );
        v1.display();
        v1.noWheels = 4;
        v1.display();
    }

}
```

```
Shwin (2 wheels) bicycle
Shwin (4 wheels) bicycle
```

A BETTER EXAMPLE

```
public class Vehicle {

    public String brand;
    private int noWheels;      // should be private...
    private String type;          // should be private...

    Vehicle( String b, int n ) {
        brand = b;
        noWheels = n;
        type = getType( n );
    }

    private String getType( int n ) {
        if ( n==1 ) return "unicycle";
        if ( n==2 ) return "bicycle";
        if ( n==3 ) return "tricycle";
        if ( n==4 ) return "car";
        return "unknown";
    }

    public void setNoWheels( int n ) {
        noWheels = n;
        type = getType( n );
    }

    public void display( ) {
        System.out.println( String.format( "%s (%d wheels) %s", brand, noWheels, type ) );
    }

    public static void main(String[] args) {
        Vehicle v1 = new Vehicle( "Shwin", 2 );
        v1.display();
        v1.setNoWheels( 4 );
        v1.display();
    }

}
```

```java
public class Vehicle {

   .
   .
   .

   public static void main(String[] args) {
      Vehicle v1 = new Vehicle( "Shwin", 2 );
      v1.display();
      v1.setNoWheels( 4 );
      v1.display();
   }

}
```

```
Shwin (2 wheels) bicycle
Shwin (4 wheels) car
```

Rules for Member Variables, or Fields:

- In general, keep them **private**
- If it is absolutely safe to let outside objects directly access a field, you may make it public. However, better **keep it private anyway**!
- Use *accessors* and |*mutators* to access the fields.

```
public void setNoWheels( int n ) {
    noWheels = n;
    type = getType( n );
}

public int getNoWheels() {
    return noWheels;
}
```

DOCUMENTATION IS IMPORTANT!

JAVADOC

GENERATES DOCUMENTATION FOR PROGRAM OR PROJECT

Javadoc

Javadocs

DOCUMENTATION OF JAVA LIBRARIES

Javadoc

- Takes comments from source program and generates HTML documentation

- Follows a strict syntax

- Reminder:
  comments in java
  look like this:

```
// this is a comment

/* this is a comment, too */

/* this
is
a
multiline comment
*/

/**
 * this is a javadoc comment
 *
 */
```

Example: Undocumented Class

```java
public class Animal {
    boolean isVaccinated;
    boolean isTattooed;
    String name;
    int age;

    Animal( String n, int a, boolean v, boolean t ) {
        name         = n;
        age          = a;
        isVaccinated = v;
        isTattooed   = t;
    }

    public void displayBasicInfo( ) {
        String v = "vaccinated";
        if ( !isVaccinated ) v = "not " + v;
        String t = "tattooed";
        if ( !isTattooed ) t = "not " + t;
        System.out.print( String.format( "%s (%d), %s, %s",
                name, age, t, v ) );
    }

    public static void main(String[] args) {
        Animal a = new Animal( "Max", 3, false, true );
        a.displayBasicInfo();
        System.out.println();
    }
}
```

Example: Ready for Javadoc (1)

```java
/**
 * Implements a super class for an animal, with a name, age, and status information
 * about vaccination and tattoo.
 * @author thiebaut
 *
 */
public class Animal {

   boolean isVaccinated;
   boolean isTattooed;
   String name;
   int age;

   /**
    * Basic constructor
    * @param n the name (String)
    * @param a the age (int)
    * @param v whether vaccinated (true = is vaccinated)
    * @param t whether tattooed (true = is tattooed)
    */
   Animal( String n, int a, boolean v, boolean t ) {
      name         = n;
      age          = a;
      isVaccinated = v;
      isTattooed   = t;
   }
```

Example: Ready for Javadoc (2)

```
    /**
     * displays all information about the animal.
     * <p>
     * Format:<br>
     * <tt>Max (3), tattooed, not vaccinated</tt>
     */
    public void displayBasicInfo( ) {
        String v = "vaccinated";
        if ( !isVaccinated ) v = "not " + v;
        String t = "tattooed";
        if ( !isTattooed ) t = "not " + t;
        System.out.print( String.format( "%s (%d), %s, %s",
                name, age, t, v ) );
    }

    /**
     * tests the class, creates an animal and displays it.
     * @param args the command line arguments (none expected)
     */
    public static void main(String[] args) {
        Animal a = new Animal( "Max", 3, false, true );
        a.displayBasicInfo();
        System.out.println();
    }

}
```

Side-Step:
Creating a Web Page

Demo

CSC212

Javadoc

Dominique Thiebaut
Dept. Computer Science

http://youtu.be/XooRQgCFJeo

Some Javadoc Tags

- **@author** [author name] - *identifies author(s) of a class or interface*.
- **@version** [version] - *version info of a class or interface.*
- **@param** [argument name] [argument description] - *describes an argument of method or constructor.*
- **@return** [description of return] - *describes data returned by method (unnecessary for constructors and void methods)*.
- **@exception** [exception thrown] [exception description] - *describes exception thrown by method.*
- **@throws** [exception thrown] [exception description] - *same as @exception.*

What about member variables and Javadoc?

```
class Toto {
    private int x;
    private boolean isValid;
    public  int age;

    Toto() {
        …
    }
    …
}
```

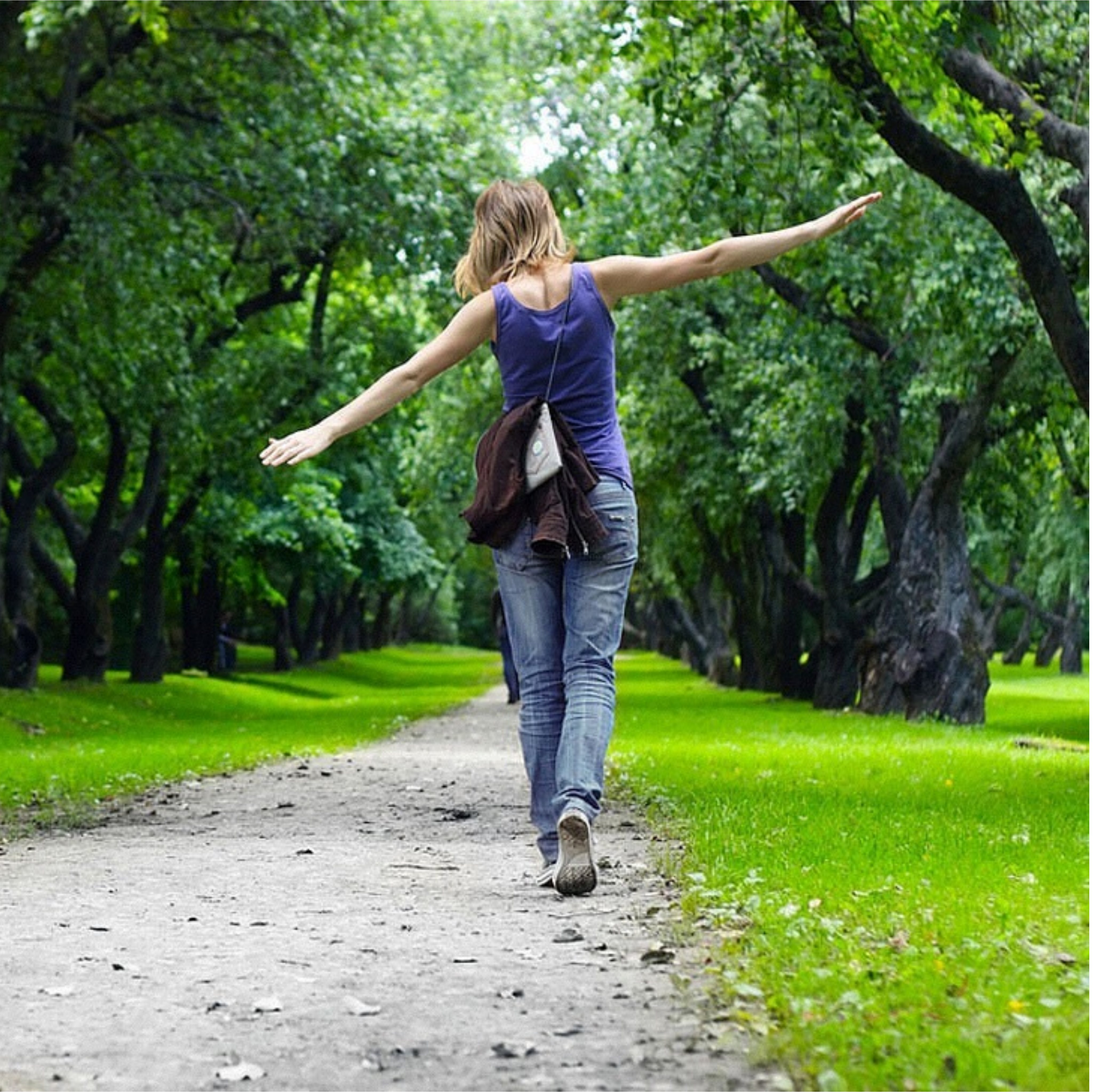
- **Java's philosophy**: you put in javadocs the information that others will need when they want to use your code.

- **private variables** are invisible and inaccessible outside the class/object, so they do not belong to javadocs.

- **public variables** are public because you want them to be accessible outside the class. Therefore you should document them in javadocs.

Javadoc Reference

http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html



Time to take a break!