6.092 Introduction to Software Engineering in Java
January (IAP) 2009

6.092: Introduction to Java

5: Packages, Containers, Using Objects

Objects

Combine data and operations on that data together (encapsulation)

Don’t need to understand the implementation, only the interface (abstraction)

Build a higher level interface from small parts

Library

```
Book[] books;
int numBooks;
String address;

void addBook(Book b) {
  books[numBooks] = b;
  numBooks++;
}
```

Library

```
void addBook(Book b);
```

class

Templates for creating objects

Define the data (*variables*) and code (*methods*) that an object has

Create an *instance* of an object with **new**

General Guidelines

Using static is rare: normally want non-static methods and variables

Class names start with CapitalLetters

Method names start with lowerCase

Library

```
Book[] books;
int numBooks;
String address;

void addBook(Book b) {
  books[numBooks] = b;
  numBooks++;
}
```

Library

```
void addBook(Book b);
```

Hiding Details

Java access control: hide unnecessary details from users

Example:

Do not change Book.borrowed variable

Call Book.borrowed(); Book.returned()

Access Control

public: Accessible by everyone

private: Accessible only by the same class

Best practice: mark everything **private** unless needed by something

```
public static void main(String[] args) {
}
```

Finding Names (scope)

General rule:

- Start in the current block { }
- Search the next enclosing block, then the next, etc …

What happens when we get to the last **class** block?

Packages

Each class belongs to a package.

Classes in the same package are automatically visible.

Classes in other packages need to be imported.

package package.name;

Importing

```
import package.name.ClassName;

import package.name.*;
```

Access Control Revisited

What if you don’t specify any access control?

Default access: accessible only from the same package

Special Packages

All classes “see” classes in the same package (no import needed)

All classes “see” classes in java.lang

Example: java.lang.String; java.lang.System

Java API

Java includes lots of classes already

Reuse these classes to avoid extra work

http://java.sun.com/javase/6/docs/api/

Putting objects in an array

- Create the array bigger than you need
- Track the next “available” slot

```
Book[] books = new Book[10];
int nextIndex = 0;

books[nextIndex] = b;
nextIndex += 1;
```

ArrayList

Provides a modifiable list

Internally implemented with arrays

- Get/put items by index
- Iterate over all items
- Add items
- Delete items

```
import java.util.ArrayList;

class ArrayListExample {
    public static void main(String[] arguments) {
        ArrayList<String> strings = new ArrayList<String>();
        strings.add("Olivier");
        strings.add("Evan");
        strings.add("Phil");

        System.out.println(strings.size());
        System.out.println(strings.get(0));
        System.out.println(strings.get(1));

        strings.set(0, "Goodbye");
        strings.remove(1);

        for (String s : strings) {
            System.out.println(s);
        }
    }
}
```

Sets

Like math: contains objects

- Is an object in the set?
- Add objects to the set
- Remove objects from the set

TreeSet: Sorted (lowest to highest)

HashSet: Unordered (pseudo-random)

```
import java.util.TreeSet;

class SetExample {
    public static void main(String[] arguments) {
        TreeSet<String> strings = new TreeSet<String>();
        strings.add("Olivier");
        strings.add("Evan");
        strings.add("Phil");

        System.out.println(strings.size());
        System.out.println(strings.first());
        System.out.println(strings.last());

        strings.remove("Evan");

        for (String s : strings) {
            System.out.println(s);
        }
    }
}
```

Maps

Stores a (*key*, *value*) pair of objects

Look up the *key*, get back the *value*

Very useful for simple “databases”

Example: Map from names to email addresses

TreeMap: Sorted (lowest to highest)

HashMap: Unordered (pseudo-random)

```
public static void main(String[] arguments) {
    HashMap<String, String> strings = new HashMap<String, String>();
    strings.put("Olivier", "koch@csail.mit.edu");
    strings.put("Evan", "evanj@mit.edu");
    strings.put("Phil", "pcm@csail.mit.edu");

    System.out.println(strings.size());

    strings.remove("Evan");

    for (String s : strings.keySet()) {
        System.out.println(s);
    }
    for (String s : strings.values()) {
        System.out.println(s);
    }
    for (Map.Entry<String, String> pairs : strings.entrySet()) {
        System.out.println(pairs);
    }
}
```

Assignment: Drawing graphics

Draw some graphics using the Java API

Use an ArrayList to hold some animated objects

Brief Introduction to Graphics