Package & Review of Class & Array

Cheng, Wei  COMP110-001  June 6, 2014
Today in COMP 110

• Package in Java

• More example on Class & Array

  – Guest Lecturer  Tom McKinney
What is Package?

• A collection of classes grouped together into a folder
  – group related classes into one unit
    • We want to divide a big project into multiple components (at many levels)
    • The highest level is a package
    • Package -> Class -> Method

• See how Eclipse is divided into packages. Each package contains several tens of classes
What is Package?

- Package can be used to resolve the name clashes of classes

- So many classes in a big project. Same name can be used in different context (or used by different programmers)
- We have a self-defined Point class in Lab 6
- There is another Point class in java.awt package
- By putting them into different packages, we can control which one to use and how it is used
To create a new package

• Create a new package in Eclipse
  – Right click your project in Package Explorer
  – New -> Package
  – Enter Package name (e.g.: myPackageA)
To group classes into a package

- Put Java class files into the package folder
  - In Eclipse, src/packageName/
  - If you are creating a new class, just right click on package and choose New -> Class.

- Each class file must have this line at the beginning of the file:

  ```java
  package Package_Name;
  ```
Existing packages in Java

• Java has provided you many packages
• We have used them in previous labs / assignments

java.lang — basic language functionality and fundamental types
all wrapper classes (Integer, Double ... ) are in this package
also String
also System (yeah. The System in System.out.println )
and the Math class - (all the static methods, abs(), log(), pow() .... )
Existing packages in Java

• Java has provided you many packages
• We have used them in previous labs / assignments

**java.util** — classes for representing and manipulating data collection

We have used *Scanner, Random, ArrayList* in this package

**java.awt** and **java.swing** — classes for building graphical user interface

We have not worked with classes in these packages directly. Instead, we interact with these classes through the class provided in lab 3. We will get more exposure in the final assignment.
Existing packages in Java

• Java has provided you many packages
• We have used them in previous labs / assignments

**java.io** — file operations (read / write files)

We will be using this very soon
Use classes in a package

• Classes in the same package can use each other directly

• If a class needs to use a class located in a different package, an import statement has to put in the beginning of the class file

  ```java
  import java.util.Scanner; // packageName.className
  ```

• The import statement should be **AFTER** the package statement if any:

  ```java
  package myPackageA;
  import java.util.*; // use * to import everything in a package
  ```

  ```java
  public class Class1 {
  .....
The only exception

- java.lang is always automatically imported
  - That’s why we can use System.out.println()....
Summary on Package

• Package corresponds to folder
• Classes in one package must have the package name declared in the first line

• To use classes in other packages, use import statement
• import statement has to be after package declaration if any
More example on Class & Array

• Here, we will implement a SortedList class:
  – It maintains a sorted list of integers
  – It has add / find / sum methods.
  – It has a merge method to merge another SortedList
public class SortedList {

    private int[] list;
    private int count;

    public SortedList( int MaxSize ) {
        list = new int[ MaxSize ];
        count = 0;
    }
}
public void add(int num) {
    if (count == list.length) {
        System.out.println("The list is full");
        return;
    }

    int insertLocation = count;
    for (int i = 0; i < count; i++) {
        if (list[i] > num) {
            insertLocation = i;
            break;
        }
    }

    for (int i = count - 1; i >= insertLocation; i--) {
        list[i + 1] = list[i];
    }

    list[insertLocation] = num;
    count++;
}

Capacity test
remember that array has fixed size

Find the insertion point

Shift everything after the insertion point to the right
SortedList

• How about the copy-once-full strategy?

```java
if (count == list.length) {
    // System.out.println("The list is full");
    // return;

    int[] newList = new int[ list.length * 2 ];
    for(int i = 0; i<count; i++)
        newList[i] = list[i];
    list = newList;
}
```
SortedList

public int get(int position) {
    return list[position];
}

public int getCount() {
    return count;
}
public int find (int num) {
    for (int i = 0; i < count; i++) {
        if (list[i] == num) return i;
    }
    return -1; // not found
}

public boolean contains (int num) {
    return find(num) >= 0;
}
public int getSum() {
    int sum = 0;
    for(int i = 0; i<count; i++) {
        sum += list[i];
    }
    return sum;
}

public void merge(SortedList L) {
    for(int i = 0; i<L.getCount(); i++) {
        this.add(L.get(i));
    }
}
Merge

- Knowing that two lists are sorted, do you have a better way to merge them?