Designing Classes Part 2

CPSC 1181 – O.O.

Jeremy Hilliker

Summer 2017

Overview

- final & static
- Constants
- enum
- Packages
- import
- Static import

final: variable

- A final variable cannot be changed once it is set
- final instance variables must be set by the time the constructor finishes
- Consider: studentID ε {1000 – 9999}

```
1   public class Student {
2       //...
3       private final int studentID;
4
5       public Student(...) {
6           //...
7           studentID = 1000 + (int)(Math.random()*8999);
8       }
9       //...
10  }
11
```

final: variable

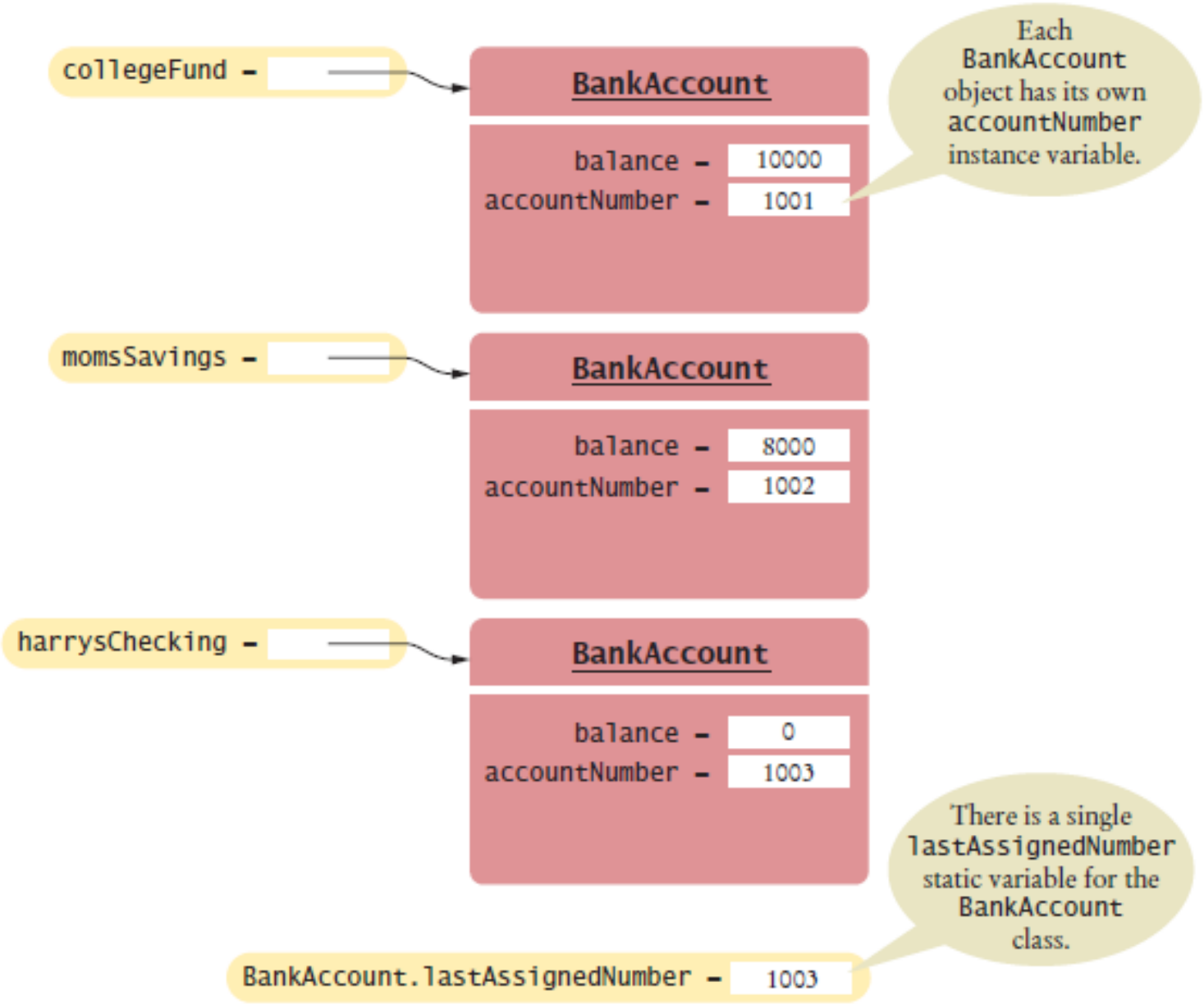
```
1   public class Error1 {
2       private final int forgotten;
3       public Error1() {
4           //... empty
5       } // <<<< compile error
6   }
7
8   public class Error2 {
9       private final int twice;
10      public Error2() {
11          twice = 1;
12          twice = 2; // <<<<< compile error
13      }
14  }
15
```

static: variable

- A static variable belongs to a class
  - Not to an object of that class
- Recall: Student ID, from 1000 to 9999

```
1   public class Student {
2       private static int nextStudentID = 1000;
3
4       private studentID;
5
6       public Student(...) {
7           ...
8           studentID = nextStudentID++;
9       }
10      ...
11  }
12
```

```
1  public class BankAccount {
2      private double balance;
3      private final int accountNumber;
4      private static int lastAccountNum = 1000;
5
6      public BankAccount() {
7          accountNumber = ++lastAccountNum;
8          balance = 0;
9      }
10 }
11
```


collegeFund
BankAccount
balance
10000
accountNumber
1001
Each BankAccount object has its own accountNumber instance variable.
momsSavings
BankAccount
balance
8000
accountNumber
1002
harrysChecking
BankAccount
balance
0
accountNumber
1003
There is a single lastAssignedNumber static variable for the BankAccount class.
BankAccount.lastAssignedNumber
1003

Constants: final static

- Owned by the class
- Cannot be changed

```
1   public class BankAccount {
2       // old way
3       public final static int TYPE_CHEQUING = 1;
4       public final static int TYPE_SAVINGS = 2;
5
6       private final int type;
7
8       public BankAccount(int aType) {
9           type = aType
10      } // Q: what is the problem with this?
11  }
12
```

Enum Types

- Owned by the class
- Cannot be changed

```
1   public class BankAccount {
2       // new way
3       public enum AccountType {
4           CHEQUING, SAVINGS
5       }
6
7       private final AccountType type;
8
9       public BankAccount(AccountType aType) {
10          type = aType
11      } // Q: why is this better?
12  }
13
```

Enum Types

```
1  public class BankAccount {
2      // new way
3      public enum AccountType {
4          CHEQUING(15.00, 0.01), SAVINGS(5.00, 0.08)
5
6          public final double monthlyFee;
7          public final double interestRate;
8
9          AccountType(double aMonthlyFee, double anInterestRate) {
10             this.monthlyFee = aMonthlyFee;
11             this.interestRate = anInterestRate;
12         }
13     } // Q: why is this better?
14
15     private final AccountType type;
16
17     public BankAccount(AccountType aType) {
18         type = aType
19     }
20 }
21
```

```java
public class CircleUtil {

    public final static int DEGREES = 360;
    public final static double RADIANS = 2 * Math.PI;
    public final static int MINUTES = 60;

    public static double percentage(double value, double max) {
        return max / double;
    }

    public static double degToRad(double deg) {
        return RADIANS * percentage(deg, DEGREES));
    }

    public static double radToDeg(double radians) {
        return DEGREES * percentage(radians, RADIANS);
    }

    public static double minutesToRad(double mins) {
        return RADIANS * percentage(mins, MINUTES);
    }
}
// to call:
double rotation = CircleUtil.minutesToRad(30);

```

Check

- The following method computes the average of an array of number:
  - public static double average(double[] values)
- Q: Why should it not be an instance method?
  - Does not access object's (instance's) state

Packages

- A set of related classes

Package	Purpose	Sample Class
java.lang	Language support	Math
java.util	Utilities	Random
java.io	Input and output	PrintStream
java.awt	Abstract Windowing Toolkit	Color
java.applet	Applets	Applet
java.net	Networking	Socket
java.sql	Database Access	ResultSet
javax.swing	Swing user interface	JButton
omg.w3c.dom	Document Object Model for XML documents	Document

Packages

- To put a class into a package, you declare it to be in that package:

  package *packageName*;

- Should be the very first line of you class file.
- If none specified, your package is in the *default* package
  - Please leave yours in this package unless told otherwise

Importing Packages

- Can refer to class by full name:
  java.util.Scanner in = new java.util.Scanner(System.in);
- Or use import:
  import java.util.Scanner;
- Or:
  import java.util.*;

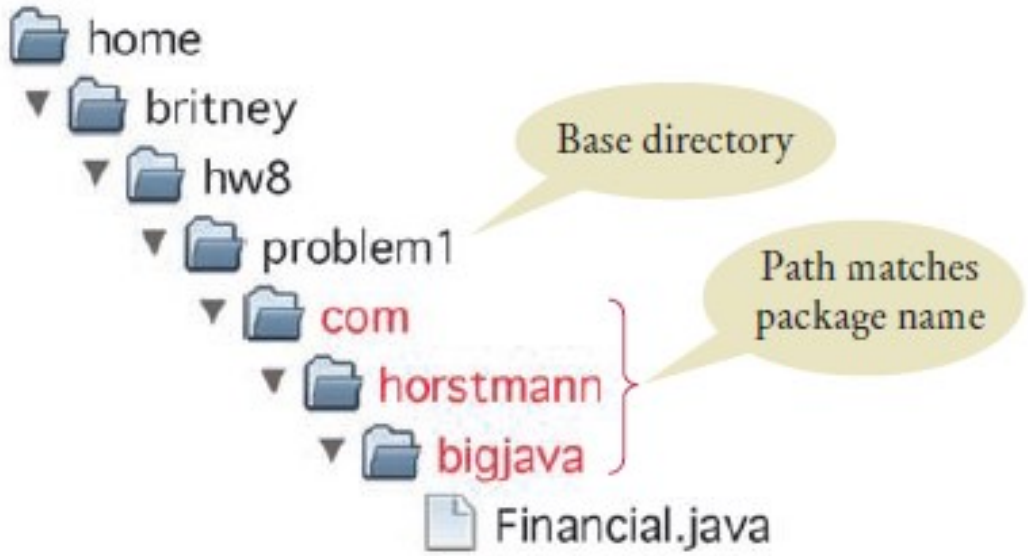
- Never need to import:
  - java.lang
  - Classes in the same package

Package Names

- Learned from “namespaces” in c++
  - Avoid name collisions:
    - java.util.Timer
    - javax.swing.Timer
- Package names should be unique:
  - From a domain name (that you own):
    - com.example.project
  - Or your email:
    - ca.mylangara.student00.project

Package and Files

- A class name must match its source file
- A package name must match its path
  - Relative to a project base name



Which of the following are packages?

a) java
b) java.lang
c) java.util
d) java.lang.Math
e) com.example.project.model
f) com.example.project.view.MainFrame

- Is a java program without package or import statements limited to using the default and java.lang packages?
  - No
- Why or why not?
  - Can use absolute classnames
  - java.util.Scanner s = new java.util.Scanner()

- Suppose your project is in the directory:
  - /home/me/cpsc1181/a5/
- You create a class:
  - q1.MysteryTester
- Where do you place it?
  - /home/me/cpsc1181/a5/**q1/MysteryTester.java**

static import

- Suppose:
  double r = Math.cos(Math.PI * theta);
- Consider:
  import static java.lang.Math.PI;
  import static java.lang.Math.cos
- Result:
  double r = cos(PI * theta);
- Problem:
  - If you do this a lot, you don't know where those things came from
  - This is exactly what "namespaces" in C were introduced to solve
    - If you "fix" everything that c programmers think is "broken" about java, you will get c
  - USE SPARINGLY!

Recap

- final & static
- Constants
- enum
- Packaged
- import
- Static import