Chapter 8

Designing Classes

Classes

- Collection of objects
- Objects are not actions
- Class names – Nouns
- Method names – Verbs
- What Makes a Good Class
  - Represent a single concept
  - Easy to understand
  - Examples
    - Concepts from math
    - Abstraction of real-life activities
    - Actors (Scanner)
    - Utility classes (Math)
    - Main (Start a program)

Not a Good Class Name

- Does not describe what the object is about
- Too complex
- Turn an action into a class
- Examples
  - You give me some examples

Cohesion and Coupling

- Cohesion
  - Classes represent a single concept
  - Public methods and constants closely related to a single concept
  - Cash Register example page 329-330
- Coupling
  - Dependencies
  - A class depends on another class if it uses objects of that class
  - May dependencies – highly coupled
  - Few dependencies – loosely coupled
  - Why matter - maintenance

Accessor, Mutator, Immutable Classes

- Mutator Method
  - Changes / modifies the object
  - In Bank Account – deposit method
- Accessor Method
  - Doesn't change / modify the object
- Immutable Class
  - No mutator methods

Side Effects

- Change an object other than the implicit parameter
- Example – transfer method of Bank Account

```
public class BankAccount
{

        other methods

        public void transfer (double amount, BankAccount otherAccount)
        {
                balance = balance - amount;
                otherAccount.balance = otherAccount.balance + amount;
        }

        other methods

}
```

Notice in red –
we are changing an account other than the this or current account.

Precondition and Post-condition

- Precondition
  - Requirement that the caller of the method must obey
  - Violate the condition – no guarantee of the result being accurate
  - Applies to method
  - Example:
    - BankAccount class – deposit method
    - Has the precondition that the amount is not negative
    - You should document the precondition
    - //@parm amount the amount of money to deposit
    - // percondition: amount >=0

Precondition and Post-condition

- Precondition
  - Way to check precondition
  - Use assert (mechanism built into Java)
  - An assertion is a condition that you believe to be true at all times in a particular program location
  - Example

```
public void deposit (double amount)
{
    assert amount > = 0;
    balance = balance + amount;
    }
```

  - Enable or disable assertions (next slide)

Precondition

- How to enable assertions
  - Run from the command line
  - java –enableassertions program name
    - The program name is the java file
    - You must include the entire data path.
    - Instead of typing out enable assertions you can use -ea
  - If the condition fails, the program terminates
  - You get a message
- Use only when testing.
- You don't use when the program is in general use.

Post Condition

- If a method has been called in accordance with its precondition, then it must assure that is postconditions are valid.
- What is a post condition.
  - The return value is computed correctly.
  - The object is in a certain state after the method is completed.
- There is not an equivalent call such as assert for post conditins.

Static Methods

- Who can tell me what it is?
  - A static method is not invoked on an object.
  - Math class
    - Why use? So you can create a method that operates on numbers.

Static Field

- You need to store values outside an particular object.
- Example:
  - Assign our bank account numbers sequentially.
  - First account number 1001, next 1002, etc.

Example

```
public class BankAccount
{
   private double balance;
   private int accountNumber;
   private static in
lastAssignedNumber;

  public BankAccount()
{
      lastAssignedNumber++;
      accountNumber =
lastAssignedNumber;
}
….
```

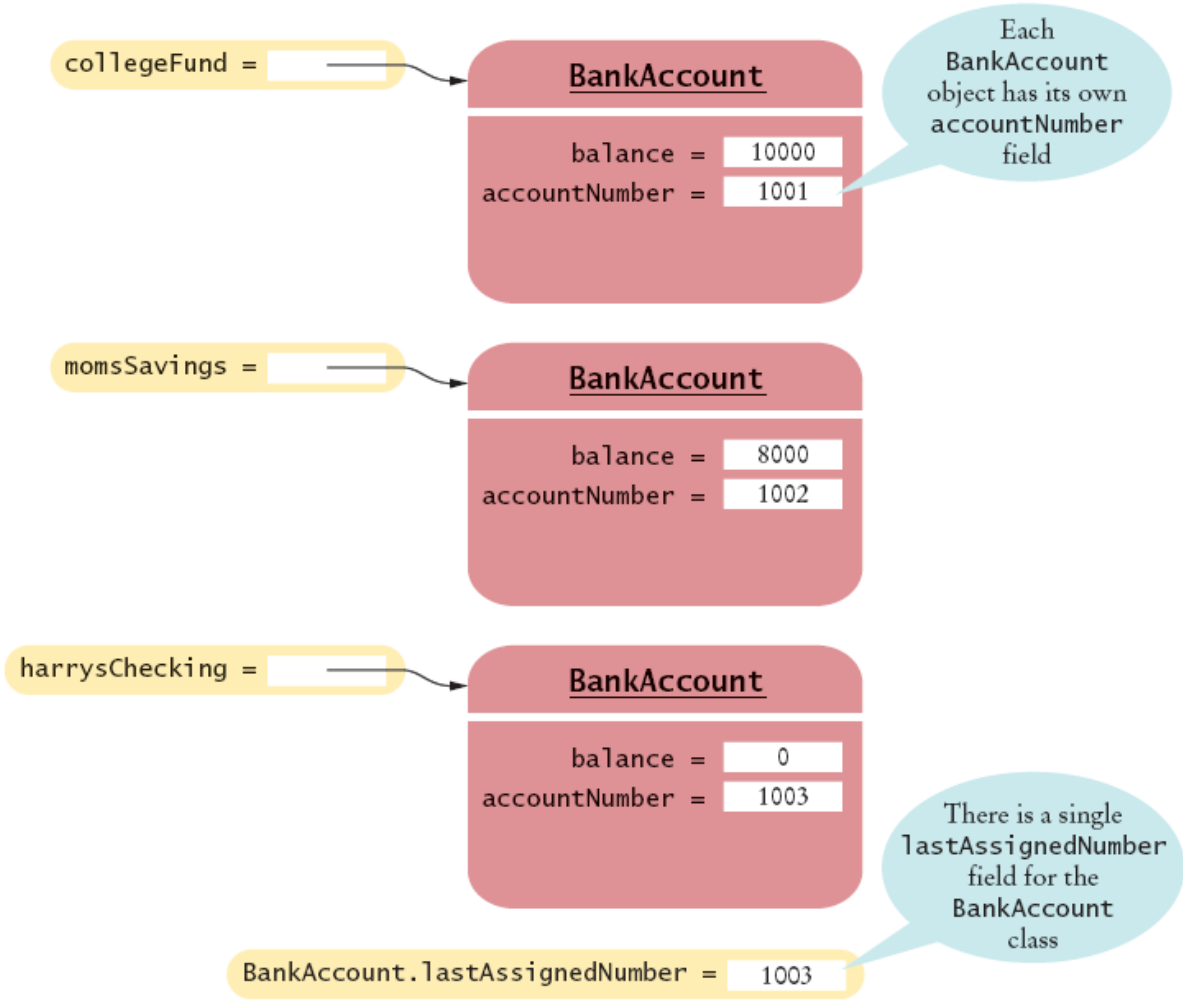

Figure 4 A Static Field and Instance Fields

Scope

- What is the scope of a variable? When does a variable live or die?
- What happens when we have overlapping scope?

Package

- Package: Set of related classes
- To put classes in a package, you must place a line
  `package packageName;`
  as the first instruction in the source file containing the classes
- Package name consists of one or more identifiers separated by periods

Important Packages in the Java Library

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
org.omg.CORBA	Common Object Request Broker Architecture	IntHolder