CSE 142

Classes and Objects in Java

Outline for Today

- Review of objects and classes
- Husky Card class design
- Class definitions in Java
- Specifications and Implementations
- Specifying methods in Java

Objects Reviewed

- Objects have properties and responsibilities
- Properties
  - Sets of ***values***
  - Have a specific ***type*** (simple or reference to an object type)
  - The current collection of property values is the object's ***state***
- Responsibilities
  - The collection of ***messages*** the object understands – what it can do
  - ***Queries*** and ***commands***

Classes Reviewed

- A collection of similar objects is called a ***class***
  - All objects in the class have the same properties and responsibilities
- Every object is an ***instance*** of some class
- The basic unit of programming in Java is a ***class definition***
  - Specifies properties and responsibilities of instances
  - Individual objects are created as needed
- Each class defines a new ***type***
  - Object properties can be references to other objects

Exercise

- **Design a class to represent a virtual Husky Card (as might be used in a simulation)**
  - **What are the properties?**
  - **What are the responsibilities?**
    Commands?
    Queries

Husky Card Design (1)

- **Properties (name, type, sample values)**

Husky Card Design (2)

- **Responsibilities (commands/queries)**

Translating this to Java

- **Class definition**

```
/** Representation of a virtual Husky Card */
public class HuskyCard {
    ...
}
```

- **Defines a class and gives it a name**
- **Between the braces { ... } we give details of**
  - ***Instance variables*: the properties of the object**
  - ***Methods*: sequences of Java code that carry out the object's responsibilities (commands and queries)**
    (In other programming languages these are sometimes called functions, procedures, or subroutines)
- **(Aside: the book uses "package" statements at the beginning of class definitions. It's not needed in small programs, so we won't use it.)**

Identifiers – Names of Things

- In the class definition

  public class HuskyCard { … }

  HuskyCard is the name of the class
- Names in Java are called ***identifiers***
  - Combination of letters, digits, underscores (_) starting with a letter ($ is also allowed, but best to avoid)
  - Must start with a letter
  - Case sensitive (abc, Abc, ABC are all different)
  - Details in the book
- You can not use a ***keyword*** or reserved word that has a special meaning in Java as an ordinary identifier

  class, public, if, for, int, double, boolean, …

Choosing Names

- Picking good names is an essential part of programming
- General rule of thumb: for names that describe classes (types), queries, and properties, use noun phrase that describes instances of the class or the property

  accountNumber, totalSales, quantityInStock, getBalance
  - Avoid cryptic, cute, or vague names

    "value" or "count" contains no useful information
- For methods, use verb phrase that describes action performed

  setBalance, deposit, withdraw, changeDate
- Capitalization – Java convention
  - Instance variables and methods begin with lower case letter
  - Class names capitalized
  - Capitalize inner words of compoundNames and CompoundClasseNames
- A class named Foo should be in a file named Foo.java

Comments

- Used to help the human reader; otherwise ignored
  - Essential to record information needed to understand the program that is not reflected directly in the code (design decisions, strategies, etc.)
- Kinds

  // the rest of the line following "//" is a comment

  /* everything after "/*" is a comment until reaching this: */

  /** special comment form for documentation ("doc comments") */
- JavaDoc tool turns the /** doc comments */ into formatted documentation web pages
- Good commenting is an art
  - Need to include essential information, but don't overdo it

Specification vs Implementation

- Specification – view of the class as seen by ***client*** code that uses instances of the class
  - Often called the interface of the class (The word "interface" also has a particular technical meaning in Java, which we will get to eventually – for now we will use it informally)
- Implementation – internal details
  - Client should not know anything about this
- Some specifications in real life
  - Automobile "user interface" – steering wheel, pedals, etc.
  - Electric power outlet

Specifying a HuskyCard

- Class: HuskyCard
- Queries
  - getName
  - getID
  - getBalance
- Commands
  - setName
  - deposit
  - withdraw
- Special "command": constructor – initialize new HuskyCard instance when it is created

HuskyCard Specification in Java

- In Java, the specification and implementation are given in a single file
- To create a class we start by writing the specification parts of methods (i.e., the operations available to client code)
- After specifying, we'll fill in the implementation details (next lecture)

Specifying Methods for Queries

- Example

```
/** Return the current balance in this HuskyCard
 *  @return the current balance in pennies. */
public int getBalance() { ... }
```

  - "public" – defines this as part of the public specification
  - "int" (or double, boolean, Color, HuskyCard, etc.) – defines the type of the value returned by this query
  - "getBalance" – the name of the method; when a getBalance message is sent to a HuskyCard object, this method will be used to carry out that responsibility

Specifying Methods for Commands

- Example

```
/** Subtract the given amount from this HuskyCard
 *  @param amount  the amount to withdraw in pennies */
public void withdraw(int amount) { ... }
```

  - "public" – same as for a query; this is part of the specification
  - "void" – special keyword to identify this as a command that does not return a value
  - "withdraw" – the name of the method
  - "int amount" is a ***parameter***, a piece of information supplied when the object is given this command
    Like the 5 in a "clap 5" message sent to an Performer

Constructors

- **Example**

```
/** Construct a new HuskyCard with an initial balance of 0
  *  @param studentName  the student's name
  *  @param IDNumber the student's ID Number */
 public HuskyCard(String studentName, int IDNumber) { … }
```

  - **Like a command, but no "void" keyword**
  - **Every time a new HuskyCard instance is created, the constructor is run**
  - **Normally used to initialize the new object's state to some sensible value**

Summary

- **Class Definitions are the unit of programming in Java**
  - **Individual objects are created as instances of these classes**
- **Specification vs Implementation**
  - **What is publicly available to client code vs what is private information hidden inside the class**
- **Specifications for class methods**
  - **Queries**
  - **Commands**
  - **Constructors – a specialized kind of command**