Introduction to Classes

COMP 110
Summer II 2012

Peter Lincoln
7/2/2012
Kinds of Types in Java

- **Primitive Types**
  - int, double, char, boolean, ...

- **Reference Types**
  - Arrays
  - Classes
Classes & Objects

- Java programs mostly consist of objects of various class types
  - OOP treats a program as a collection of objects that interact by means of actions

- To create an object in a program, we must provide a definition of objects, called a class
  - Definition includes how objects behave and kinds of information they maintain

- Objects can represent objects in the real world
  - E.g. cars, houses, employee records
- Objects can represent abstract concepts
  - E.g. colors, shapes, words
Classes & Objects

- Your programs have used some objects and invoked their methods
  - String myString = "Hello!";
  - myString.length();
  - myString.toLowerCase();
  - System.out.println(myString);
  - System.exit(0);

- Now it’s time to define your own simple classes and uses objects and methods of those classes
Class

- A class is the definition of a kind of object
- A blueprint for constructing specific objects

**Class Name:** Automobile

**Data:**
- amount of fuel
- speed
- license plate

**Methods (actions):**
- accelerate
- decelerate
  - How: Press on brake pedal.
Objects & Instantiation

Object Name: sueCar
amount of fuel: 14 gallons
speed: 0 miles per hour
license plate: “SUES CAR”

Object Name: patsCar
amount of fuel: 10 gallons
speed: 55 miles per hour
license plate: “135 XJK”

Object Name: jimsCar
amount of fuel: 2 gallons
speed: 75 miles per hour
license plate: “582 KFS”

Object Name: firemanCar
amount of fuel: 7 gallons
speed: 35 miles per hour
license plate: “451 F”

Objects of the class Automobile
Objects can have Data.

Object Name: jimsCar

amount of fuel: 2 gallons
speed: 75 miles per hour
license plate: “582 KFS”

Classes do not have data, they define the types of data that objects have

Class Name: Automobile

Data:
- amount of fuel
- speed
- license plate

Methods (actions):
- accelerate
- decelerate
  - How: Press on brake pedal.
Class Name: Automobile

Data:
- amount of fuel
- speed
- license plate

Methods (actions):
- accelerate
  How: Press on gas pedal.
- decelerate
  How: Press on brake pedal.

Automobile
- fuel: double
- speed: double
- license: double

+ accelerate(double pedalPressure) : void
+ decelerate(double pedalPressure) : void
Class Files in Java

- Each Java class definition goes in its own, separate .java file

- Class name and filename prefix must be the same
  - Automobile → Automobile.java
  - Student → Student.java
  - StringFun → StringFun.java
Class Files in Java

- What happens when you compile a .java file?
  - .java file is compiled into a .class file
    - .class file contains Java bytecode
    - Same filename prefix between .java and .class
      - StringFun.java → StringFun.class

- Class files are compiled independently
  - You can compile a Java class before you have a program that uses it
  - Eclipse automatically compiles your code as soon as you hit save
Class Design Example

- Suppose we wanted to design a class to represent a student...
UML for Student Class

Class Name

Data Kinds
(Instance Variables)

Methods
(Actions)

Student

- name : String
- classYear : int
- major : String
- creditsSum : int
- gpaSum : double

+ getName() : String
+ getMajor() : String
+ printData() : void
+ increaseYear() : void
+ addGrade(int credits, double gpa) : void
+ calcGPA() : double
public class Student {
    public String name;
    public int classYear;
    public String major;
    public int creditsSum;
    public double gpaSum;

    public String getName() {
        return name;
    }

    // ...

    public void addGrade(int credits, double gpa) {
        creditsSum += credits;
        gpaSum += gpa;
    }

    public double calcGPA() {
        return gpaSum / creditsSum;
    }
}
Creating an instance of Student

- Create an object jack of class Student
  - `Student jack = new Student();`

- Allocate memory for an object and return that address

- Assign memory address of object to variable

- Create an object

- `Scanner keyboard = new Scanner(System.in);`
Instance Variables

- **Syntax**
  - public *Type* *variableName*;

- **public**: No restriction on how to use the variable (more details later)
  - public is not generally recommended for instance variables

- **Type**: The type of variable’s value

- **Instance variables use the same naming conventions as other variables**: camelCase

- **Examples**
  - public *String* *name*;
  - public *int* *classYear*;
  - public *double* *gpa*;
  - public *String* *major*;

7/3/2012

COMP 110 - Summer II 2012
public class Student {

    public String name;
    public int classYear;
    public String major;
    public int creditsSum;
    public double gpaSum;

    public String getName() {
        return name;
    }

    // ...

    public void addGrade(int credits, double gpa) {
        creditsSum += credits;
        gpaSum += gpa;
    }

    public double calcGPA() {
        return gpaSum / creditsSum;
    }

}
Using public Instance Variables Outside the Class Definition

- `public class StudentDemo {
    
    public static void main(String[] args) {
        Student jack = new Student();
        jack.name = "Jack Smith";
        jack.major = "Computer Science";

        System.out.println(jack.name + " is majoring in " + jack.major);

        Student apu = new Student();
        apu.name = "Apu Nahasapeemapetilon";
        apu.major = "Biology";

        System.out.println(apu.name + " is majoring in " + apu.major);
    }
}

- `jack.name` and `apu.name` are two different instance variables because they belong to two different objects: `jack` and `apu`
Methods

- There are two kinds of methods
  - Methods that return a value
    - `myString.substring(…)`
    - `myString.indexOf(…)`
  - Methods that don’t return a value
    - `System.out.println(…)`
Method Definition

- Syntax for methods without arguments
  - public Type methodName()
    {
      Method_Body
    }
  
  - public void methodName()
    {
      Method_Body
    }

- public: No restriction on how to use the method (more details later)
- Type: The type of value that the method returns
- void: Indicates that the method returns no value

- The body of the method, Method_Body, must be enclosed in {}
Method Definition Examples

```
public String getName()
{
    return name;
}

public void increaseYear()
{
    classYear++;
}
```

Returns String

Return Type

Returns Nothing
Method Body

- **Is a block**
  - As usual, a block can contain multiple statements

- **Example**
  - `public void printData()`
    ```java
    {
    System.out.println("Name: " + name);
    System.out.println("Major: " + major);
    System.out.println("GPA: " + gpa);
    }
    ```
The return Statement

- Syntax
  - return Expression;

- A method that returns a value must have at least one return statement
- Terminates the method and returns the value of Expression back to the caller
  - The type of Expression’s value must match the return type of the method

- Example
  - public String getMajor()
    {
      return major;
    }
The return Statement

- The return statement does not need to be at the end of the method body
  - `public String getClassYear()`
    ```java
    { 
      if (classYear == 1) 
        return "Freshman";
      else if (classYear == 2) 
        return "Sophomore";
      // ... 
      else 
        return "Unknown";
    }
    ```
  - It will still terminate the method in this case
The return Statement

- Can also be used in methods that return nothing
  - Terminates the method

- Syntax
  - `return;`

- Example
  - `public void increaseYear()`
    ```java
    {
        if (classYear >= 4)
            return;
        classYear++;  // Terminates the method
    }
    ```
Calling Methods That Return Nothing

- Syntax for methods without arguments
  - `objectName.methodName();`

- Use them as Java statements
  - `jack.increaseYear();`
  - `System.out.println(“Hello World!”);`
Calling Methods That Return a Value

- Syntax for methods without arguments
  - `objectName.methodName()`

- Use them as Java expressions or statements
  - String `maj = jack.getMajor();`
  - `System.out.println("Jack’s GPA: " + jack.calcGPA());`
Syntax Summary

- **Class Declaration**
  - public class ClassName
    
    | Instance_Variable_List |
    |------------------------|
    | Method_Definition_List |

- **Instance Variable Declaration**
  - public Type variableName;

- **Method Definition**
  - public Type methodName()
    
    | Method_Body |
    |-------------|

  - public void methodName()
    
    | Method_Body |
    |-------------|

- **Calling Methods w/o Parameters**
  - objectName.methodName();
Questions?
Logistics

Next:

Lab 5: Bubble Sort