COMP 110
Introduction to Programming

Fall 2015
Time: TR 9:30 – 10:45
Room: AR 121 (Hanes Art Center)

Jay Aikat
FB 314, aikat@cs.unc.edu

Previous Class

• What did we discuss?
Today

• Announcements
  • Lab 2: Due Wednesday, Oct 21 @ 11:55 PM

• Midterm solution: Sakai ➔ Resources

• Arrays

• Classes and Methods

2D Arrays

• Arrays having more than one index are often useful
  — Tables
  — Grids
  — Board games

<table>
<thead>
<tr>
<th></th>
<th>0: Open</th>
<th>1: High</th>
<th>2: Low</th>
<th>3: Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Apple Inc.</td>
<td>99.24</td>
<td>99.85</td>
<td>95.72</td>
<td>98.24</td>
</tr>
<tr>
<td>1: Walt Disney Co.</td>
<td>21.55</td>
<td>24.20</td>
<td>21.41</td>
<td>23.36</td>
</tr>
<tr>
<td>2: Google Inc.</td>
<td>333.12</td>
<td>341.15</td>
<td>325.33</td>
<td>331.14</td>
</tr>
<tr>
<td>3: Microsoft Corp.</td>
<td>21.32</td>
<td>21.54</td>
<td>21.00</td>
<td>21.50</td>
</tr>
</tbody>
</table>
Declaring and Creating 2D Arrays

• Two pairs of square brackets means 2D
  – int[][] table = new int[3][4];

• or
  – int[][] table;
  – table = new int[3][4];

Declaring and Creating 2D Arrays

• Array (or 1D array) gives you a list of variables
  – int[] score = new int[5] gives you score[0], score[1], ..., score[5]

• 2D array gives you a table of variables
  – int[][] table = new int[3][4];
Using a 2D Array

• We use a loop to access 1D arrays

```java
for (int i = 0; i < 5; i++) {
    scores[i] = keyboard.nextInt();
    scoreSum += scores[i];
}
```

Using a 2D Array

• We use nested loops for 2D arrays

```java
int[][] table = new int[4][3];
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 3; j++) {
        table[i][j] = i * 3 + j;
        System.out.println(table[i][j]);
    }
}
```
Multidimensional Arrays

- You can have more than two dimensions
  - `int[][][] cube = new int[4][3][4];`
- Use more nested loops to access all elements
  - for (int i...)
    - for (int j...)
      - for (int k...)

Today's topic: classes

```java
public class Program1 {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```
Classes and Objects

• Java programs (and programs in other object-oriented programming languages) consist of objects of various class types
  – Objects can represent objects in the real world
    • Automobiles, houses, students
  – Or abstract concepts
    • Colors, shapes, words
• When designing a program, it’s important to figure out what is a class/object in your program

Class

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

<table>
<thead>
<tr>
<th>Class Name: Automobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data:</td>
</tr>
<tr>
<td>amount of fuel</td>
</tr>
<tr>
<td>speed</td>
</tr>
<tr>
<td>license plate</td>
</tr>
<tr>
<td>Methods (actions):</td>
</tr>
<tr>
<td>accelerate:</td>
</tr>
<tr>
<td>Action: increase speed</td>
</tr>
<tr>
<td>decelerate:</td>
</tr>
<tr>
<td>Action: decrease speed</td>
</tr>
</tbody>
</table>
Objects (Instances)

- Instances of the class Automobile

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

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

Object Name: emmettsCar
amount of fuel: 5 gallons
speed: 88 miles per hour
license plate: “OUTATIME”

UML (Universal Modeling Language)

UML diagram or Class diagram

Automobile
- fuel: double
- speed: double
- licensePlate: String

+ accelerate(double pedalPressure): void
+ decelerate(double pedalPressure): void
Class Files and Separate Compilation

• Each Java class definition goes in its own .java file
• For a class named ClassName, you should save the file as ClassName.java
• Student.java shall and must include the class Student

Class Files and Separate Compilation

• What happens when you compile a .java file?
  – .java file gets compiled into a .class file
    • Contains Java bytecode (instructions)
    • Same filename except for .class instead of .java
• You must compile a Java class before you or a program can use it
• You can send the .class file to people who use it, without revealing your actual code
Class Student

- A general UML class specification

<table>
<thead>
<tr>
<th>Class Name: Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Name</td>
</tr>
<tr>
<td>- Year</td>
</tr>
<tr>
<td>- GPA</td>
</tr>
<tr>
<td>- Major</td>
</tr>
<tr>
<td>- Credits</td>
</tr>
<tr>
<td>- GPA sum</td>
</tr>
<tr>
<td>+ getName</td>
</tr>
<tr>
<td>+ getMajor</td>
</tr>
<tr>
<td>+ printData</td>
</tr>
<tr>
<td>+ increaseYear</td>
</tr>
</tbody>
</table>

  Action: increase year by 1

Class Student

- A detailed UML class specification (in Java)

<table>
<thead>
<tr>
<th>Class Name: Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>- name: String</td>
</tr>
<tr>
<td>- year: int</td>
</tr>
<tr>
<td>- gpa: double</td>
</tr>
<tr>
<td>- major: String</td>
</tr>
<tr>
<td>- credits: int</td>
</tr>
<tr>
<td>- gpaSum: double</td>
</tr>
<tr>
<td>+ getName(): String</td>
</tr>
<tr>
<td>+ getMajor(): String</td>
</tr>
<tr>
<td>+ printData(): void</td>
</tr>
<tr>
<td>+ increaseYear(): void</td>
</tr>
</tbody>
</table>
Defining a Class

```java
public class Student {
    public String name;
    public int classYear;
    public double gpa;
    public String major;
    // ...
    public String getMajor()
    {
        return major;
    }
    public void increaseYear()
    {
        classYear++;         
    }
}
```

Creating an Object

- **Syntax**
  - `ClassName objectName = new ClassName();`
- **What does the statement do?**
  - The computer will create a new object, and assign its memory address to `objectName`
  - `objectName` is sometimes called a class type variable
    - It is a variable of class type `ClassName`
- **Why do we need new?**
  - So we know `ClassName()` is not executing a method but creating an object
Creating an Object

Create an object *jack* of class *Student*

```
Student jack = new Student();
```

Create an object *keyboard* of class *Scanner*

```
Scanner keyboard = new Scanner(System.in);
```

Instance Variables

- Data defined in the class are called *instance variables*

```java
public String name;
public int classYear;
public double gpa;
public String major;
```

- *type: int, double, String…*
- *variable names*
- *public: no restrictions on how these instance variables are used (more details later – public is actually a bad idea in some cases)*
Using Instance Variables Inside a Class

```java
public class Student {
    public String name;
    public int classYear;
    public double gpa;
    public String major;
    // ...
    public String getMajor()
    {
        return major;
    }
    public void increaseYear()
    {
        classYear++;
    }
}
```

Any instance variables can be freely used inside the class definition.

Using `public` Instance Variables Inside a Class

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

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

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

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

```java
public class Student {
    public String name;
    public int classYear;
    public double gpa;
    public String major;
    // ...

    public String getMajor() {
        return major;
    }
    public void increaseYear() {
        classYear++;
    }
}
```

**Methods**

- Two kinds of methods
  - Methods that return a value
    - Examples: String’s `substring()` method, String’s `indexOf()` method, String’s `charAt()` method, etc.
  - Methods that return nothing
    - Example: `System.out.println()`
- “Return” means “give back”
  - A method can give back a value so that other parts of the program can use it, or simply perform some actions
Defining Methods that Return a Value

- Method heading: keywords
  - public: no restriction on how to use the method (more details later)
  - Type: the type of value the method returns
- Method body: statements executed
  - Must be inside a pair of brackets
  - Must have a return statement

```
public String getMajor()
{
    return major;
}
```

return Statement

- A method that returns a value must have at least one return statement
- Terminates the method, and returns a value
- Syntax:
  - return expression;
- expression can be any expression that produces a value of type specified by the return type in the method heading
Methods that Return a Value

As usual, inside a block (defined by braces), you can have multiple statements

```java
public String getClassYear()
{
    if (classYear == 1)
        return "Freshman";
    else if (classYear == 2)
        return "Sophomore";
    else if ...
}
```

Calling Methods that Return a Value

- Object, followed by dot, then method name, then ()
  – `objectName.methodName();`
- Use them as a value of the type specified by the method’s return type

```java
Student jack = new Student();
jack.major = "Computer Science";

String m = jack.getMajor(); // Same as String m = "Computer Science"

System.out.println("Jack’s full name is " + jack.getName());
// Same as System.out.println("Jack’s full name is " + "Jack Smith");
System.out.println("Jack’s major is " + m);
```
Defining Methods That Return Nothing

- Method heading: keywords
  - public: no restriction on how to use the method (more details later)
  - void: the method returns nothing

- Method body: statements executed when the method is called (invoked)
  - Must be inside a pair of brackets

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

Methods that Return Nothing

```java
public void printData()
{
    System.out.println("Name: " + name);
    System.out.println("Major: " + major);
    System.out.println("GPA: " + gpa);
}
```
Calling Methods that Return Nothing

- Object, followed by dot, then method name, then ()
  - The same as a method that returns a value
  - `objectName.methodName();`
- Use them as Java statements

```java
Student jack = new Student();
jack.classYear = 1;
jack.increaseYear();
System.out.println("Jack's class year is " + jack.classYear);
```

return Statement in void Methods

- Can also be used in methods that return nothing
- Simply terminates the method
- Syntax:
  - `return;`

```java
public void increaseYear()
{
    if (classYear >= 4)
        return;
    classYear++;
}
```
Next class

• More on Classes and Methods