Java Introduction

Jasleen Kaur

Fall 2011

Java references

- You can
  - Simply use Java references on the web
    - Will suffice for this course
  - Purchase Addison-Wesley's Java Backpack Reference Guide
    - Available cheap from several sources
  - Purchase a Java text
    - Expensive option
    - Recommended only if you plan on continuing in CS
Let's try the sample code provided

- I'm not using jGRASP (but Java command-line)
  - You'll get to use jGRASP in recitation this Friday
- Let's run
  - With valid input
  - With invalid input
- Let's introduce code errors
  - ;
  - Remove definition of a variable
  - Make spelling mistake

Let's next scan the code

- A lot seems familiar
  - Assignment is still: =
    
    ;
    
    if
    
    while
    
    for
    
    //
    
    /* ... */
- Some is different
  - Definition of variables
  - Overall structure
    
    public static void main(String[] args)
    
    {
    
    }

© Copyright: Jasleen Kaur. 2011.

Introduction to Programming
Java object model

- You create a project
  - A project can have many classes
    - Each class can have several subroutines
  - Each class is in a separate file
    - .java (written by you)
    - .class (~ Java compiled code)

- Every program is a class
  - `public class pgmName {` Same as file name
  - `...` Outside world can use
  - `public static void main(String[] args) {` name For parameters
  - `...`

Java basics

- Comments: same as JS
  ```
  //     /* ... */
  ```

- Naming conventions: same as JS
  - Letter + { letter / digit / _}* Case matters
  - Avoid reserved keywords

- Data types:
  - Integer: int
  - Real: double
  - Boolean: boolean
  - Character: char
  - String: String Not a primitive type; class names start with upper case
Java basics

- ; – same as JS

Creating variables
  - type name;
  - type name = initial value;

```
int x = 5;
double z = 3.4;
boolean test = true;
char ch = 'A'; // single quotes for characters
String name = “Jasleen”; // double quotes here
```

- Strongly-typed!
  - Cannot change type of a variable

Assignment:
  - Variable = expression;
    ```
x = 3 + 4 * 5;
```
  - Mostly same rules, other than for ...

Defining and using “constants”
  - final type name = value;
    ```
final double PI = 3.1416;
final int STDWEEK = 40;
```
  - Naming convention: all upper case letters
  - Used just like variables in expressions, except:
    ```
PI = 3; // Illegal -> can’t be changed
```
## Expressions & types

- **int op int** ➞ int
  
<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 + 4</td>
<td>7</td>
</tr>
<tr>
<td>3 * 4</td>
<td>12</td>
</tr>
<tr>
<td>3 / 4</td>
<td>0</td>
</tr>
</tbody>
</table>

- **double op double** ➞ double
  
<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5 + 1.0</td>
<td>4.5</td>
</tr>
<tr>
<td>3.5 * 2.0</td>
<td>7.0</td>
</tr>
<tr>
<td>3.0/4.0</td>
<td>0.75</td>
</tr>
</tbody>
</table>

- **Mixed expressions** ➞ double
  
<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 + 4.0</td>
<td>7.0</td>
</tr>
<tr>
<td>3 * 4.0</td>
<td>12.0</td>
</tr>
<tr>
<td>3 / 4.0</td>
<td>0.75</td>
</tr>
<tr>
<td>3.0 / 4</td>
<td>0.75</td>
</tr>
</tbody>
</table>

## Type casting

- Can have integer treated as real (and vice versa):

  ```java
  int x = 3;
  (double) x ➞ has value 3.0
  (int) 3.14 ➞ has value 3 (truncated)
  ```
Simple I/O

- **Output:**
  - Needs some “magic” lines (explained later)
    ```java
    import java.io.*;
    ... throws IOException
    System.out.println(stringArg);
    ```
    - Prints stringArg and adds a newline character (‘\n’) at end
    ```java
    System.out.print(stringArg);
    ```
    - Prints stringArg without a newline at the end

- **Input:**
  - “Magic” lines:
    ```java
    Scanner kb = new Scanner(System.in);
    ```
  - Input:
    ```java
    line = kb.nextLine();
    num1 = Integer.parseInt(line); // convert to int
    ```

Selection – same as JS

```java
if (boolean expression)
{
    ...
}
else
{
    ...
}
```

- **Same rules about need for:** `{ } ;`
Iteration

- **While loops**
  
  ```java
  while (boolean expression)
  {
    ...
  }
  ```

- **For loops**
  
  - With previously defined iterator variable
    ```java
    for ( i = 0; i < n; i++ )
    {
      ...
    }
    ```
  
  - Iterator variable defined (and exists!) only in loop
    ```java
    for ( int i = 0; i < n; i++ )
    {
      ...
    }
    ```

    *break; // get out of loop
    *continue; // skip to next iteration of loop*