CS201 Introduction to Java Programming

Introduction to Java Applications

Chengyu Sun
California State University, Los Angeles

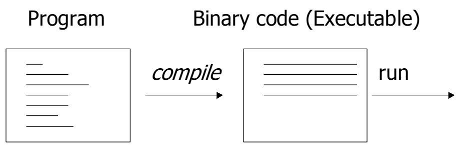
Overview

- Java programming language
- Anatomy of a Java application
- Programming Environment – *JBuilder X*

Programming Languages

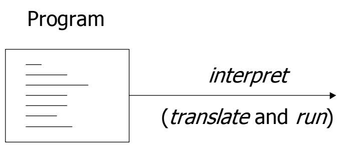
- Machine languages
  - 0100010111001 …
- Assembly languages
  - `load`, `store`, `move`, `cmp`, …
- High-level languages
  - Basic, C/C++, Fortran, Ada, Java …

Compile



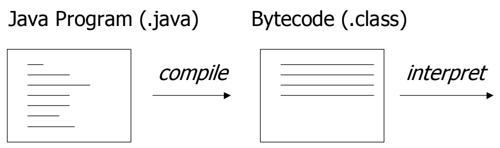
- C/C++, Fortran, Ada, Assembly …
- Compiler and Assembler

Interpret



- BASIC, scripting languages …
- Interpreter

Compile and Interpret



- Java
  - Compiler – `javac`
  - Interpreter (Java Virtual Machine) – `java`
  - Compiler + Interpreter – `JBuilder`, `NetBeans`, …

Advantages of Java

- Elegant OO language design
- Huge standard library
- Good documentation
- JVM
  - Portability
  - Web-oriented features
  - Safe

Some facts about Java

- Who – James Gosling and co.
- Where – SUN Microsystems
- When – early 1990's
- Different forms of Java program
  - Application
  - Applet

Example: Sum100

- Calculate the sum of 1, 2, … , 99, and 100

Sum100.java

```
/**
 * Calculate 1+2+3+...+100
 */

public class Sum100 {

  public static void main( String args[] )
  {
    int sum = 0;
    for( int i=1 ; i <= 100 ; ++i )
      sum = sum + i;

    // output
    System.out.println( sum );
  }
} // end of class Sum100
```

Program Structure – Comments

```
/**                                         Comments
 * Calculate 1+2+3+...+100
 */

public class Sum100 {

  public static void main( String args[] )
  {
    int sum = 0;
    for( int i=1 ; i <= 100 ; ++i )
      sum = sum + i;

    // output                               Comments
    System.out.println( sum );
  }
} // end of class Sum100                    Comments
```

Comments

- Description of certain program functions
- Ignored by Java compiler
- Can appear anywhere of the program

```
/* a comment */          // another comment

/* a                     /*
  multiple-line           * a better looking
  comment                 * multiple-line comment
*/                        */
```

Program Structure – Class

Class Header

```
public class Sum100 {

  public static void main( String args[] )
  {
    int sum = 0;
    for( int i=1 ; i <= 100 ; ++i )
      sum = sum + i;

    System.out.println( sum );
  }
}
```

Class

Class

- Class header (declaration)
  - `public` – access modifier
  - `class`
  - *Class name*
    - User specified
    - must be the same as the file name
    - E.g. *Sum100* and *Sum100.java*
- Class body
  - Enclosed in a pair of {}

Class Names and Names in General

- Rules
  - Must start with a letter
  - Cannot conflict with any language keywords/symbols
  - Case-sensitive
- Conventions
  - Class names start with a upper-case letter
  - Method/variable names start with a lower-case letter
  - Multiple word concatenated directly, except for *constants*

Program Structure – Method

```
public class Sum100 {

  public static void main( String args[] )
  {
    int sum = 0;
    for( int i=1 ; i <= 100 ; ++i )
      sum = sum + i;

    System.out.println( sum );
  }
}
```

Method Header

Method

Method

- Method header (declaration)
  - `public` – access modifier
  - `static`
  - `void`
  - *Method name*
    - User specified
    - `main` – a special method, where the execution of a Java application begins
  - *Arguments* – enclosed in a pair of ()
- Method body
  - Enclosed in a pair of {}

Program Structure – Statements

```
public class Sum100 {

  public static void main( String args[] )
  {
    int sum = 0;                              statement
    for( int i=1 ; i <= 100 ; ++i )
      sum = sum + i;                          statement

    System.out.println( sum );                statement
  }
}
```

Statements

- "Sentences" in a programming language
  - Generally ends with a semicolon, except some control statements
  - May consist of other statements

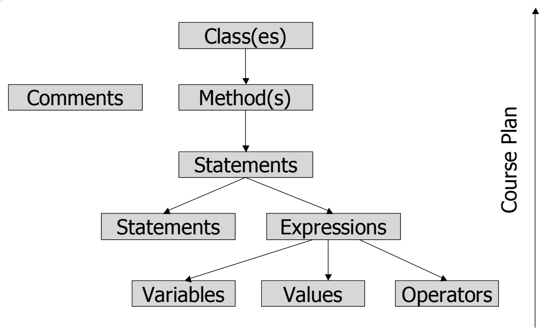
Program Structure – Expressions

```
public class Sum100 {

  public static void main( String args[] )
  {
    int sum = 0;
    for( int i=1 ; i <= 100 ; ++i )
      sum = sum + i;                    expression

    System.out.println( sum );
  }
}
```

Expressions

- A combination of variables, values (literals), and operators that evaluates to a single value

Program Structure – Summary



Basic Program Structure

JBuilder X

- Developed by Borland
- Three different versions
  - Foundation – freely downloadable
  - Developer – available on all lab machines
  - Enterprise – we don't need it

Create a New Project

- `File` → `New Project …`
  - Step 1
    - Name
    - Directory – somewhere you can find it
  - Step 2
  - Step 3
    - Javadoc fields

Add a Java File to the Project

- Right click `<Project Source>` and select `New` → `Class …`
  - Class name
  - Package – leave it empty
- Edit the Java file

Compile and Run the Project

- Compile
  - Click the `Make Project` button on the toolbar
- Run
  - Click the `Run Project` button on the toolbar
  - Runtime Configuration
    - Main class

Exercise

- Welcome2
  - p40 [D&D]