Building Java Programs

Chapter 1
Lecture 1-1: Introduction; Basic Java Programs

reading: 1.1 - 1.3
What is CSE?

- **Computer Science**
  - The systematic study of algorithmic processes that create, describe, and transform information. -- Wikipedia
    - Algorithm: effective method for solving problem expressed as finite sequence of instructions. -- Wikipedia
  
- **Many subfields**
  - Graphics, Computer Vision
  - Artificial Intelligence, Robotics
  - Scientific Computing
  - Databases, Data Mining
  - Computational Linguistics, Natural Language Processing...

- **Computer Engineering**
  - Overlap with CS and EE; emphasizes hardware
What is programming?

- **program**: A set of instructions to be carried out by a computer.

- **program execution**: The act of carrying out the instructions contained in a program.

- **programming language**: A systematic set of rules used to describe computations in a format that is editable by humans.
Some influential ones:

- FORTRAN
  - science / engineering
- COBOL
  - business data
- LISP
  - logic and AI
- BASIC
  - a simple language
Some modern languages

- **procedural languages**: programs are a series of commands
  - **Pascal** (1970): designed for education
  - **C** (1972): low-level operating systems and device drivers
- **functional programming**: functions map inputs to outputs
  - **Lisp** (1958) / **Scheme** (1975), **ML** (1973), **Haskell** (1990)
- **object-oriented languages**: programs use interacting "objects"
  - **Smalltalk** (1980): first major object-oriented language
  - **C++** (1985): "object-oriented" improvements to C
    - successful in industry; used to build major OSes such as Windows
  - **Java** (1995): designed for embedded systems, web apps/servers
    - Runs on many platforms (Windows, Mac, Linux, cell phones...)
    - The language taught in this textbook
Why Java?

- Relatively simple
- Object-oriented
- Pre-written software
- Platform independent
- Widely used
  - #1 in popularity ie http://www.tiobe.com
Basic Java programs with println statements

reading: 1.2 - 1.3
Compiling/running a program

1. Write it.
   - **code** or **source code**: The set of instructions in a program.

2. Compile it.
   - **compile**: Translate a program from one language to another.
   - **byte code**: The Java compiler converts your code into a format named *byte code* that runs on many computer types.

3. Run (execute) it.
   - **output**: The messages printed to the user by a program.

```
Hello.java

---jGRASP exec: java Hello
Hello, World!

---jGRASP: operation completed
```
A Java program

public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
        System.out.println();
        System.out.println("This program produces");
        System.out.println("four lines of output");
    }
}

- Its output:
  Hello, world!

  This program produces
  four lines of output

- **console**: Text box into which the program's output is printed.
Structure of a Java program

```java
public class name {
    public static void main(String[] args) {
        statement;
        statement;
        ...
        statement;
    }
}
```

- Every executable Java program consists of a **class**, that contains a **method** named **main**, that contains the **statements** (commands) to be executed.
Names and identifiers

- You must give your program a name.

```java
public class IrishPoetry {

- Naming convention: capitalize each word (e.g. MyClass_Name)
- Your program's file must match exactly (IrishPoetry.java)
  - includes capitalization (Java is "case-sensitive")

- **identifier**: A name given to an item in your program.
  - must start with a letter or _ or $
  - subsequent characters can be any of those or a number
    - **legal**: _myName    TheCure    ANSWER_IS_42    $bling$
    - **illegal**: me+u    49ers    side-swipe    Ph.D's
```
Keywords

- **keyword**: An identifier that you cannot use because it already has a reserved meaning in Java.

  
  abstract  default  if  implements  private  this
  boolean  do  implements  protected  throw
  break  double  import  public  throws
  byte  else  instanceof  return  transient
  case  extends  int  short  try
  catch  final  interface  static  void
  char  finally  long  strictfp  volatile
  class  float  native  super  while
  const  for  new  switch synchronized
  continue  goto  package

Copyright 2010 by Pearson Education
Syntax

- **syntax**: The set of legal structures and commands that can be used in a particular language.
  - Every basic Java statement ends with a semicolon ;
  - The contents of a class or method occur between { and }

- **syntax error (compiler error)**: A problem in the structure of a program that causes the compiler to fail.
  - Missing semicolon
  - Too many or too few { } braces
  - Illegal identifier for class name
  - Class and file names do not match
  ...

Copyright 2010 by Pearson Education
Syntax error example

```java
public class Hello {
    pooblic static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

**Compiler output:**

Hello.java:2: <identifier> expected
    pooblic static void main(String[] args) {
          ^
Hello.java:3: ';' expected
    }
          ^
2 errors

- The compiler shows the line number where it found the error.
- The error messages can be tough to understand!
**System.out.println**

- A statement that prints a line of output on the console.
  - pronounced "print-linn"
  - sometimes called a "println statement" for short

- Two ways to use **System.out.println**:
  - `System.out.println("text");`
    Prints the given message as output.
  - `System.out.println();`
    Prints a blank line of output.
**Strings**

- **string**: A sequence of characters to be printed.
  - Starts and ends with a " quote " character.
    - The quotes do not appear in the output.
  - Examples:
    - "hello"
    - "This is a string. It's very long!"

- **Restrictions**:
  - May not span multiple lines.
    - "This is not a legal String."
  - May not contain a " character.
    - "This is not a "legal" String either."
Escape sequences

- **escape sequence**: A special sequence of characters used to represent certain special characters in a string.
  - \t tab character
  - \n new line character
  - \" quotation mark character
  - \\ backslash character

- **Example:**
  ```java
  System.out.println("\\hello\\nhow\tare \"you\"?\\\\");
  ```

- **Output:**
  ```java
  \hello
  how are "you"?
  ```
Questions

- **What println statements will generate this output?**

  This quote is from Irish poet Oscar Wilde:

  "Music makes one feel so romantic
  - at least it always gets on one's nerves -
  which is the same thing nowadays."

- **What println statements will generate this output?**

  A "quoted" String is
  'much' better if you learn
  the rules of "escape sequences."

  Also, "" represents an empty String.
  Don't forget: use " instead of "!
  '' is not the same as ""
• println statements to generate the output:
  System.out.println("This quote is from");
  System.out.println("Irish poet Oscar Wilde:");
  System.out.println();
  System.out.println("\"Music makes one feel so romantic\"");
  System.out.println("- at least it always gets on one's nerves -");
  System.out.println("which is the same thing nowadays.\"");

• println statements to generate the output:
  System.out.println("A \"quoted\" String is");
  System.out.println("'much' better if you learn");
  System.out.println("the rules of \"escape sequences.\"");
  System.out.println();
  System.out.println("Also, \"\" represents an empty String.");
  System.out.println("Don't forget: use \\\" instead of \" !");
  System.out.println("'\' is not the same as \"");
Questions

- What is the output of the following `println` statements?

```java
System.out.println("	a	b	c");
System.out.println("
\\");
System.out.println("\"");
System.out.println("\"");
System.out.println("\"");
System.out.println("C:\nin\the downward spiral");
```

- Write a `println` statement to produce this output:

```
/ \ // \ \ /// \ \
```
Answers

• Output of each println statement:

```
  a  b  c
```

```
  / \\
 /  \
/   \
/    \
```

C:
he downward spiral

• println statement to produce the line of output:

```java
System.out.println("/ \ // \ \ \ // \ \ \ \ // \ \ \ \ \ // \ ");
```