Lecture 1: Overview of Java

What is java?

- Developed by Sun Microsystems (James Gosling)
- A general-purpose object-oriented language
- Based on C/C++
- Designed for easy Web/Internet applications
- Widespread acceptance

Java Features (1)

- **Simple**
  - fixes some clumsy features of C++
  - no pointers
  - automatic garbage collection
  - rich pre-defined class library
- **Object oriented**
  - focus on the data (objects) and methods manipulating the data
  - all functions are associated with objects
  - almost all data types are objects (files, strings, etc.)
  - potentially better code organization and reuse

Java Features (2)

- **Interpreted**
  - java compiler generate byte-codes, not native machine code
  - the compiled byte-codes are platform-independent
  - java bytecodes are translated on the fly to machine readable instructions in runtime (Java Virtual Machine)
- **Portable**
  - same application runs on all platforms
  - the sizes of the primitive data types are always the same
  - the libraries define portable interfaces

Java Features (3)

- **Reliable**
  - extensive compile-time and runtime error checking
  - no pointers but real arrays. Memory corruptions or unauthorized memory accesses are impossible
  - automatic garbage collection tracks objects usage over time
- **Secure**
  - usage in networked environments requires more security
  - memory allocation model is a major defense
  - access restrictions are forced (private, public)

Java Features (4)

- **Multithreaded**
  - multiple concurrent threads of executions can run simultaneously
  - utilizes a sophisticated set of synchronization primitives (based on monitors and condition variables paradigm) to achieve this
- **Dynamic**
  - java is designed to adapt to evolving environment
  - libraries can freely add new methods and instance variables without any effect on their clients
  - interfaces promote flexibility and reusability in code by specifying a set of methods an object can perform, but leaves open how these methods should be implemented
  - can check the class type in runtime

Getting Started: (1)

(1) Create the source file:

- open a text editor, type in the code which defines a class (*HelloWorldApp*) and then save it in a file (*HelloWorldApp.java*)
- file and class name are case sensitive and must be matched exactly (except the `.java` part)

Example Code: HelloWorldApp.java

```
/**
 * The HelloWorldApp class implements an application
 * that displays "Hello World!" to the standard output
*/
public class HelloWorldApp {
  public static void main(String[] args) {
     // Display "Hello World!"
     System.out.println("Hello World!");
  }
}
```

★ Java is CASE SENSITIVE!

Getting Started: (2)

(2) Compile the program:

- compile HelloWorldApp.java by using the following command:

```
javac HelloWorldApp.java
```

it generates a file named HelloWorldApp.class

☹ **'javac' is not recognized as an internal or external command, operable program or hatch file.**

```
javac: Command not found
```

if you see one of these errors, you have two choices:

1) specify the full path in which the `javac` program locates every time. For example:

```
C:\j2sdk1.4.2_09\bin\javac HelloWorldApp.java
```

2) set the PATH environment variable

Getting Started: (3)

(3) Run the program:

- run the code through:

```
java HelloWorldApp
```

- Note that the command is `java`, not `javac`, and you refer to `HelloWorldApp`, not `HelloWorldApp.java` or `HelloWorldApp.class`

☹ **`Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorldApp`**

if you see this error, you may need to set the environment variable CLASSPATH.

Basic Syntax

- **Case Sensitivity -** Java is case sensitive, which means identifier **Hello**and **hello** would have different meaning in Java.
- **Class Names -** For all class names the first letter should be in Upper Case.

  If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.

  Example *class MyFirstJavaClass*
- **Method Names -** All method names should start with a Lower Case letter.

  If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.

  Example *public void myMethodName()*
- **Program File Name -** Name of the program file should exactly match the class name.

  When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match your program will not compile).

  Example: Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as '*MyFirstJavaProgram.java*'
- **public static void main(String args[]) -** Java program processing starts from the main() method which is a mandatory part of every Java program.

Java Identifiers

- All Java components require names. Names used for classes, variables and methods are called identifiers.
- In Java, there are several points to remember about identifiers. They are as follows:
  - All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).
  - After the first character identifiers can have any combination of characters.
  - A key word cannot be used as an identifier.
  - Most importantly identifiers are case sensitive.
  - Examples of legal identifiers: age, $salary, _value, __1_value
  - Examples of illegal identifiers: 123abc, -salary

Java Modifiers

- Like other languages, it is possible to modify classes, methods, etc., by using modifiers. There are two categories of modifiers:
- **Access Modifiers:** default, public , protected, private
- **Non-access Modifiers:** final, abstract, strictfp
- We will be looking into more details about modifiers in the next section.

Language basics (1)

- Data types
  - 8 primitive types:
    - boolean, byte, short, int, long, float, double, char
  - Class types, either provided by Java, or made by programmers
    - String, Integer, Array, Frame, Object, Person, Animal, …
  - Array types
- Variables
  - *dataType identifier [= Expression]:*
  - Example variable declarations and initializations:

```
int x;     x=5;
boolean b = true;
Frame win = new Frame();
String x = “how are you?”;
```

```
int[] intArray;
intArray = new int[2];
intArray[0] = 12;
intArray[1] = 6;
Person pArray = new Person[10];
```

Language basics (2)

- Flow of control
  - if, if-else, if-else if
  - switch
  - for, while, do-while
  - break
  - continue