Object Declaration

- Every object used must be declared
- Syntax:
  `<class name> <object name>;

  - `<class name>`: the name of the class to which the object belongs
  - `<object name>`: the name of the object (any valid identifier)

- **Identifier**: any sequence of letters, digits, underscores, and dollar signs with the following limitations:
  - Must begin with a letter
  - Cannot contain any spaces or other “white space”
  - Cannot be a Java “reserved” word (aka “keyword”)

- **Java Convention on Identifiers**:
  - First letter lowercase
  - First letter of subsequent words uppercase

- **Reserved Word**: an Identifier that is used for a specific purpose and cannot be used for any other purpose.

  - Example of some of Java’s Reserved Words:
    ```
    public          private          protected
    import          class            new
    static          void             byte
    short           int              long
    float           double           boolean
    final           return           while
    if              do              for
    ```
Object Creation

- No objects are actually created by a declaration (with declaration, only an Identifier used to refer to an object is created)

- Use the ‘new’ command. Syntax:

  \[ \text{<object name>} = \text{new} \ <\text{class name}> \ (<\text{arguments}>) \]

  - \text{<object name>}: the name of the declared object
  - \text{<class name>}: the name of the class to which the object belongs
  - \text{<arguments>}: sequence of values passed to the method

Examples of Object Declaration:

Student dave;
Noisemaker clapper;
Ship battleship;

Examples of Object Creation:

dave = new Student (4.0, 1234);
crapper = new Noisemaker ();
battleship = new Battleship (numPegs, xPos, yPos, dir);
Message Sending

- Once an object has been created, messages can be sent to it

- Syntax:

  \texttt{<object name>.<method name> (<arguments>);}

  - \texttt{<object name>}: name of a declared object
  - \texttt{.}: the “dot notation” gives relation to the items on either side of the dot
  - \texttt{<method name>}: name of a method of the object
  - \texttt{<arguments>}: sequence of values passed to the method

Examples of Message Sending:

dave.setGPA (2.5);

clapper.makeNoise (decibelLevel);

battleship.insertHit ();
Program Components

Three (3) main parts:
1. **Comments**
2. Import statements
3. Class declarations

### Comments

- **Uses:**
  1. State the purpose of the program
  2. Explain the meaning of code
  3. Give other explanations to help programmers understand the program

- **Syntax:**
  
  /* ANY text between slash-asterisk and asterisk-slash */

  OR

  // ANY text following two slashes to the end of the line

- All programs should contain a **Header Comment** containing the following information:
  1. Program Title
  2. Author
  3. Course (including section number)
  4. Date Written (or Due Date)
  5. Description of Program

- Comments are NOT required to run a program. However, they are indispensable in writing easy to understand code. (You will lose points if your programs do NOT contain adequate comments.)

- Excessive comments can hurt more than help in understanding code.
Program Components

Three (3) main parts:
1. Comments
2. Import statements
3. Class declarations

Import Statements

- Classes are grouped into “Packages”
- To use a class from a Package, the class must be “imported” into the program. Syntax:

  import <package name>.<class name>;

  - import: a reserved word indicating a class is to be imported
  - <package name>: the name of the package to which the class belongs
  - <class name>: the name of the class to be imported

- With subclasses, use multiple dot notations
  For example: import java.awt.image.ColorModel;

- To import more than one class from a package, use the asterisk notation. Syntax:

  import <package name>*;

- When the asterisk notation is used, ALL of the classes (or subclasses) of a particular package (or super class) will be imported.

- Java Convention: all package names are lowercase.
Program Components

Three (3) main parts:
1. Comments
2. Import statements
3. Class declarations

Class Declaration

- Syntax:

```
class <class name>
{
   <class member declarations>
}
```

- `class`: a reserved word indication the declaration of a class
- `<class name>`: the name of a class (any valid identifier)
  
  Java convention: class names start with a capital letter and each subsequent word in the class name also has a capital letter

- `<class member declarations>`: a sequence of class member declarations
  
  `class member`: a data value or a method

- A program can (and usually does) have more than one class, but only one class will be designated the “main” class.

- Typically, the application and the main class have the same name

- The main class must define a method called main. This method is executed FIRST when the Java application is executed.
Method Declaration

❖ Syntax:

```
<modifiers> <return type> <method name> (<arguments>)
{
    <method body>
}
```

- `<modifiers>`: sequence of terms designating different kinds of methods
- `<return type>`: type of data value returned by a method
- `<method name>`: name of the method (any valid identifier)
- `<arguments>`: sequence of values passed to a method
- `<method body>`: sequence of instructions
Edit-Compile-Run Cycle

Three (3) steps:
1. Step 1: Type in the program using an editor and save it
2. Step 2: Compile the source file
3. Step 3: Execute the bytecode file using an Interpreter

Step 1: Type in the program using an editor and save it

◆ Examples of editors:
  • Code Warrior
  • JavaWorks
  • vi
  • emacs

◆ Save the entered code with the following filename syntax:

<name of main class>.java

◆ The resultant is a source file written in a “high level language” (HLL)

◆ Examples of high level languages;
  • Java
  • C
  • C++
  • Pascal
  • BASIC
  • Fortran

◆ Machines (i.e., computers) can only understand machine language (written in binary). Machine language is a “low level language” (LLL).
Edit-Compile-Run Cycle

Three (3) steps:
1. Step 1: Type in the program using an editor and save it
2. Step 2: Compile the source file
3. Step 3: Execute the bytecode file using an Interpreter

Step 2: Compile the source file

- A compiler translates the HLL into a LLL called bytecode
  (Code Warrior contains a compiler)

- The bytecode file that is generated is titled as follows:
  <name of source file>.class

- The whole source file is compiled at once

- Compilers can detect Compilation Errors (aka “Syntax Errors”)
  - Compilation Errors: errors resulting from the source code containing text that does not obey the rules of the language
  - Examples of Compilation Errors:
    - Mismatched parantheses ( ( ) )
    - Missing punctuation (e.g., no semi-colon at the end of statements)
    - Misspelled reserved words

- The compiler will NOT generate a bytecode file if compilation errors exist in the source file.

- Most good compilers give detailed error messages when identifying the compilation errors
Edit-Compile-Run Cycle

Three (3) steps:
1. Step 1: Type in the program using an editor and save it
2. Step 2: Compile the source file
3. Step 3: Execute the bytecode file using an Interpreter

Step 3: Execute the bytecode file using an Interpreter

- The interpreter executes instructions one line at a time
- The interpreter can detect Execution Errors (aka “Run-Time Errors”)
  - Execution Errors: errors occurring during the execution of the instructions
  - Examples of Execution Errors:
    - Dividing by zero
    - Using undeclared objects/data values
    - Null pointers

Editor \[\rightarrow\] Compiler \[\rightarrow\] Interpreter \[\rightarrow\] Running Program

source file \[\rightarrow\] bytecode file

---

Editor

Compiler

Interpreter

Running Program

bytecode file

source file