Object-Oriented Programming

Java and Hello World

Computer Science and Technology
United International College

Review

- What is a class?
- What is an object?
- Association
- Multiplicity
- Aggregation
- Composition
- Inheritance
- Polymorphism
- Interface

Outline

- Computer languages and Java
- History of Java
- Java Architecture
- Java Virtual Machine (JVM)
- Garbage collection
- Java Development environment
- First Java Program – HelloWorld.java

Computer Languages

- Machine languages:
  - 10011110111100001111111 [Binary style]
- Low-level languages:
  - Assembly language:
  - add ax, bx [Intel x86 style]
- High-level: human-friendly languages:
  - Structured programming languages [Fortran, C, Pascal];
  - C++ , a combination of structured programming & object-oriented;
  - Java, pure object-oriented language;
  - C#, similar to Java.

History of Java

- Developed by Sun Microsystems.
- Sun was bought by Oracle in 2010.
  - 1996: Java version 1.0;
  - … Java 1.1, 1.2, 1.3, 1.4;
  - 2004: Java version 1.5, name changed to Java 5;
  - … Java 6, 7, 8.

  One new release every 6 months now:
  - September 2017: Java 9;
  - March 2018: Java 10;
  - September 2018: Java 11 (long term support);
  - March 2019: Java 12.

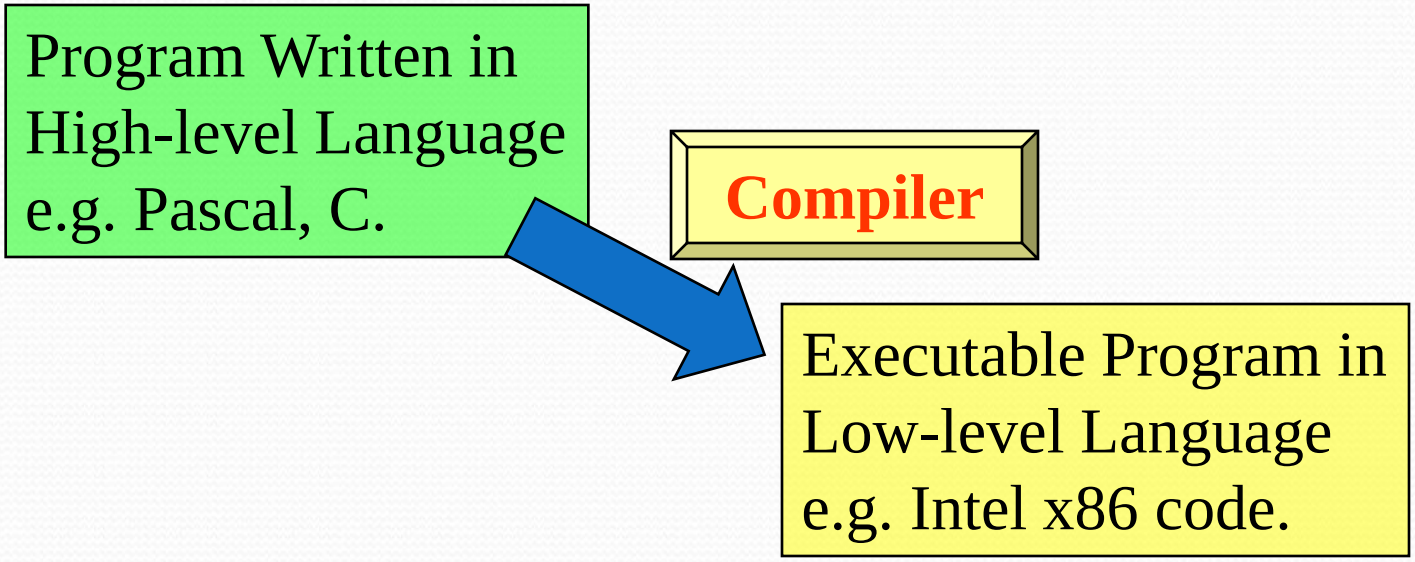
Features of Java

- Simple
- Object-oriented
- Platform neutral
- Robust, secure, scalable.

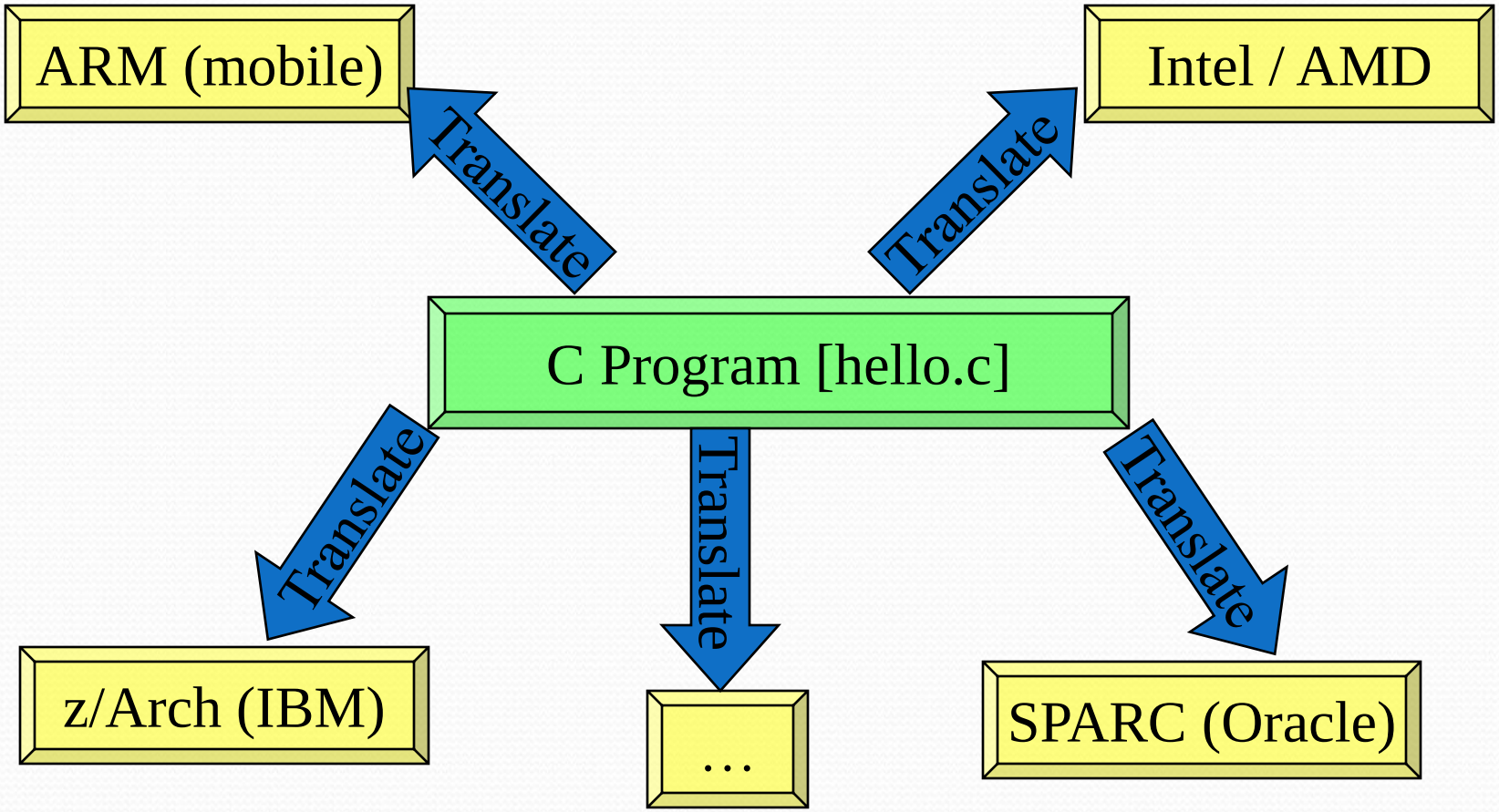
  E.g.: programmers don’t need to care about memory allocation and release.
- Java is everywhere!

Compiler

- ***Compiler***: a program that translates a high-level language program into an equivalent low-level language program.
- This translation process is called ***compiling.***



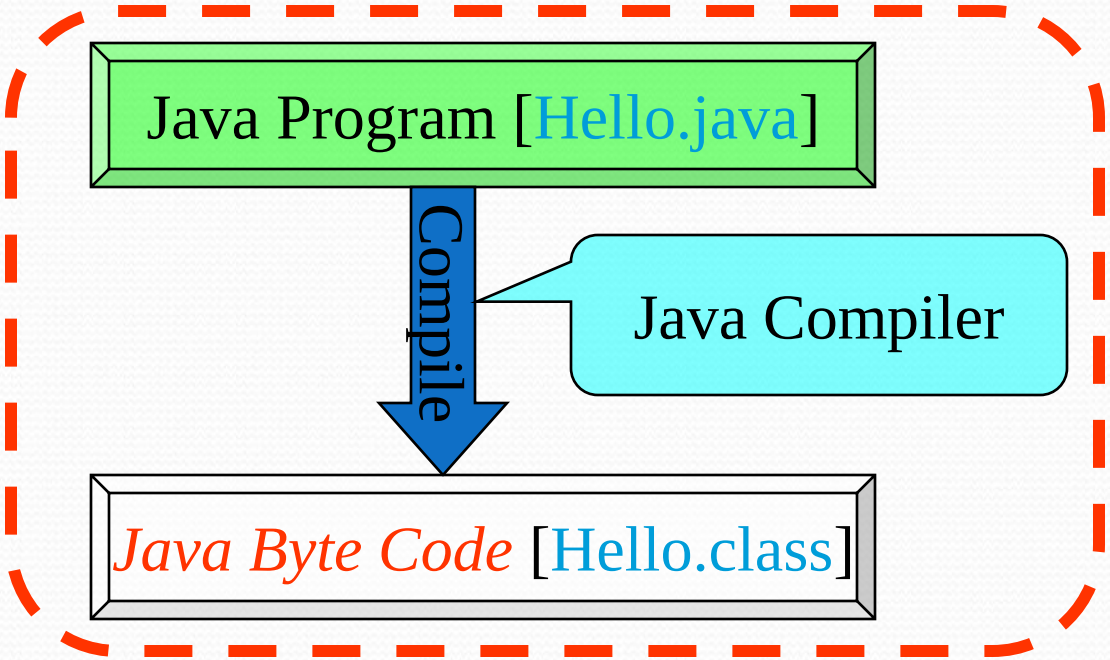
Really Cross Platform?



Pitfalls

- Store and use different *compiled* versions.
- Availability of compiler(s) is a must.
- Compiler compatibility problems.
- Re-compile all versions after an update.

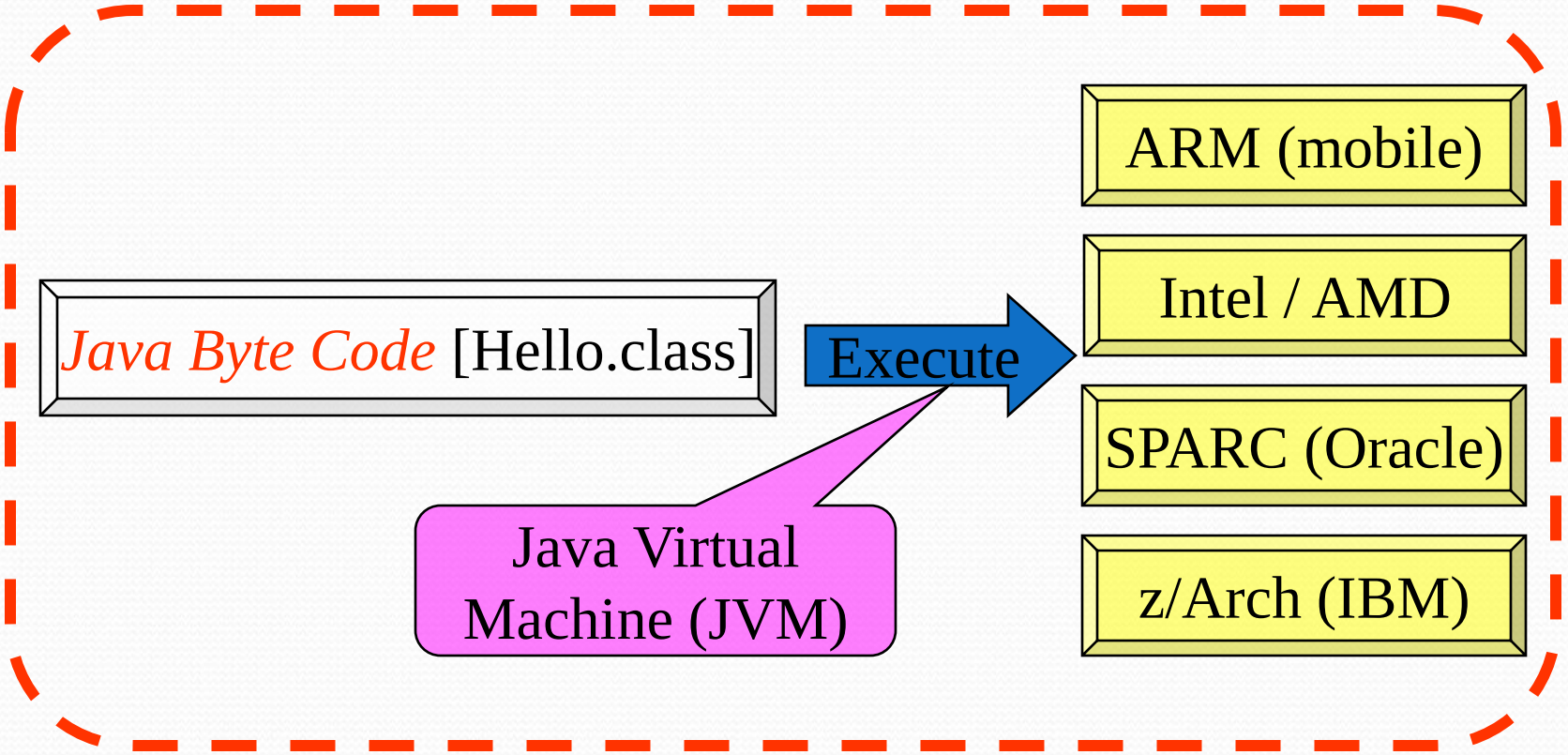
Java Compilation Model



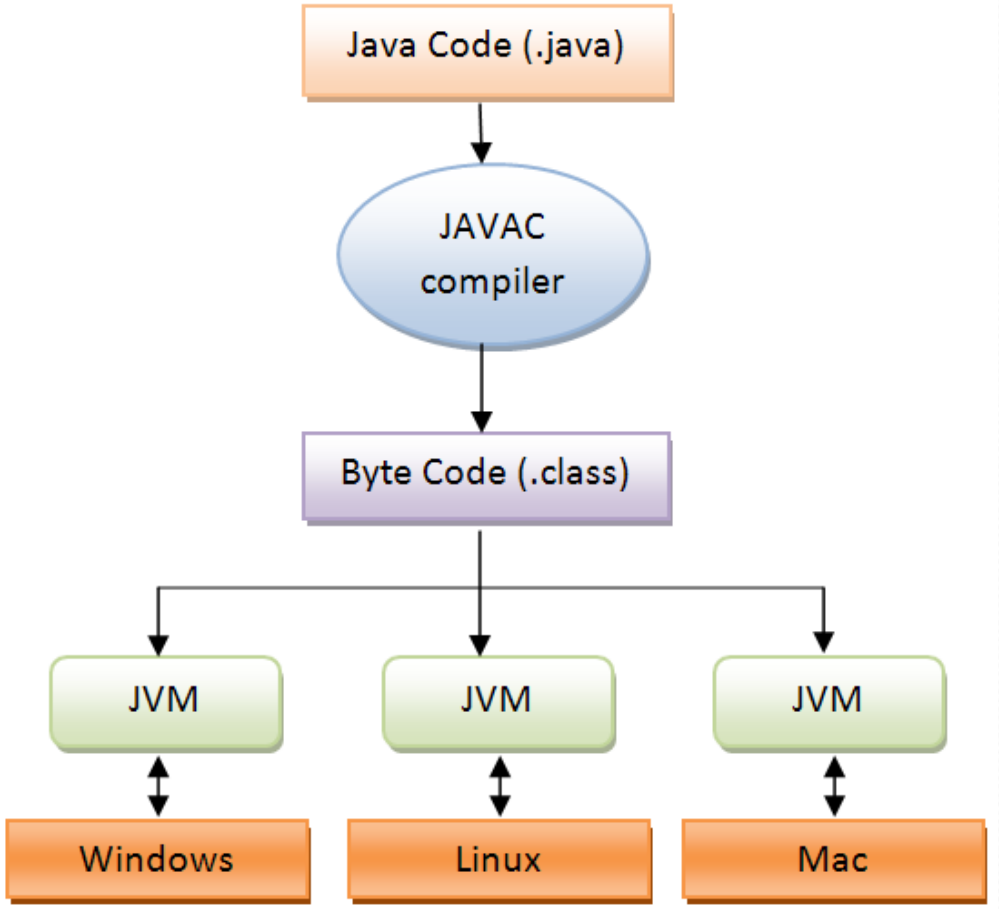
We do this once

Java Compilation Model

Someone else creates this for us.



Java Compilation Model



Byte-Code and the Java Virtual Machine

- Java compiler translates Java programs into ***byte-code***, a language for the ***Java Virtual Machine.***

- Once compiled to ***byte-code***, a Java program can be used on ***any*** computer that has a JVM, making it very portable.

Write-Once → Run-Anywhere

The heart of Java

- **Java Virtual Machine**
  - Reads “byte code” and executes it.
- **Garbage collection**
  - Manages memory automatically;
  - Included in the JVM.

Two Java Environments

- JRE
  - Java Runtime Environment = JVM + Java libraries.
  - For users to run byte-code programs on their computer.
- JDK
  - Java Development Kit = JRE + Java compiler + other development tools.
  - For developers to write Java software, create byte-code, and test it.

Three Java Editions

- Java Micro Edition (JavaME)
  - Very small Java edition for smart cards, pages, phones, and set-top boxes.
  - Subset of the standard Java libraries aimed at limited size and processing power.
- Java Standard Edition (JavaSE)
  - The basic platform, which this course will cover.
- Java Enterprise Edition (JavaEE)
  - For business applications, web services, mission-critical systems.
  - JSP, Servlets, JDBC, EJB3.0, Struts – Spring – Hibernate.

Example: the Java SE 8u161 JDK

- Java programming language;
- Standard Edition: includes all libraries for PCs;
- Version 8;
- Update 161: the Java environment is updated regularly to fix bugs;
- Java Development Kit: includes tools for programmers, such as the Java compiler.

The First Program

- Install the Java SE JDK on your computer: https://www.java.com/en/
- Install Eclipse: https://www.eclipse.org/downloads/ (select "Eclipse IDE for Java Developers" when installing);
- See the link "How To Install Eclipse on Your Own Computer" on iSpace for more information.

Running a Java Program

- Start Eclipse and create a workspace: a folder that will contain all your Java projects.
- Create a Java project.
- Add a new class to your project, name it HelloWorld.
- Type the code of the HelloWorld class: see next slide.
- Building (compiling Java into byte code) is automatic.
- Run the program, which executes the byte code on a JVM.

Hello World Program

```
public class HelloWorld {

   /* The HelloWorld Program
      -------------------
      Illustrates a simple program displaying
      a message.
   */

   public static void main (String[] args) {
     System.out.println("HelloWorld!");
   }
}
```

Type all carefully and save it to a file named HelloWorld.java.

The First Java Program

Java program source files (.java) contain definitions of classes.

```
public class HelloWorld {

  /* The HelloWorld Program
     -------------------
     Illustrates a simple program displaying
     a message.
  */

  public static void main (String[] args) {
    System.out.println("HelloWorld!");
  }
}
```

The First Java Program

Curly braces pair enclose a block of code, class HelloWorld here.

```
public class HelloWorld {

  /* The HelloWorld Program
     -------------------
     Illustrates a simple program displaying
     a message.
  */

  public static void main (String[] args) {
    System.out.println("HelloWorld!");
  }
}
```

Don’t miss me!

The First Java Program

Curly braces pair enclose a block of code, method main( ) here.

```
public class HelloWorld {

   /* The HelloWorld Program
      ------------------
      Illustrates a simple program displaying
      a message.
   */

   public static void main (String[] args) {
      System.out.println("HelloWorld!");
   }
}
```

Don't miss me!

The First Java Program

```
public class HelloWorld {

   /* The HelloWorld Program
      ------------------
      Illustrates a simple program displaying
      a message.
   */

   public static void main (String[] args) {
      System.out.println("HelloWorld!");
   }
}
```

This is a block of comments, for human, not for computer.

It explains to you what happens.

The First Java Program

/* and */ pair encloses a comment block.

```
public class HelloWorld{

   /* The HelloWorld Program
      -------------------
      Illustrates a simple program displaying
      a message.
   */

   public static void main (String[] args) {
     System.out.println("HelloWorld!");
   }
}
```

Don’t miss me!

Comments

- `// Comment single line`
- ```
  /* Comment
     multiple
     lines
    */
  ```
- ```
  /**
  *
  **/
  ```

The First Java Program

This is a method of the class HelloWorld, named main( )

```
public class HelloWorld {

   /* The HelloWorld Program
      -------------------
      Illustrates a simple program displaying
      a message.
   */

   public static void main (String[] args) {
      System.out.println("HelloWorld!");
   }
}
```

The First Java Program

Standard properties of the main( ) method.

```
public class HelloWorld {

   /* The HelloWorld Program
      ---------------------
      Illustrates a simple program displaying
      a message.
   */

   public static void main (String[] args) {
      System.out.println("HelloWorld!");
   }
}
```

The First Java Program

A statement (instruction) to display a message.

```
public class HelloWorld {

   /* The HelloWorld Program
      ----------------------
      Illustrates a simple program displaying
      a message.
   */

   public static void main (String[] args) {
      System.out.println("HelloWorld!");
   }
}
```

The First Java Program

```
public class HelloWorld {

   /* The HelloWorld Program
      -------------------
      Illustrates a simple program displaying
      a message.
   */

   public static void main (String[] args) {
      System.out.println("HelloWorld!");
   }
}
```

After every statement, there must be a semi-colon!

What does this String[] args do?

Java Programming

- A Java program consists of ***objects*** that interact with one another by means of actions (called ***methods***).
- Other high-level languages (C, Fortran) have procedures, functions, and subprograms.
  - These are called ***methods*** in Java.
  - All programming constructs in Java, including *methods*, appear in an ***object*** (which is an ***instance*** of a ***class***).

Java Programming

- A Java ***application program*** is a class with a method named **`main`**.
  - When a Java application program is run, the JVM automatically executes the byte-code for the method named **`main`**.
  - All Java application programs start with the **`main`** method.
  - Your Java software is going to contain many classes, but only one class has a **`main`** method.

Java Programming

- Application programs may use:
  - a windowing interface (GUI): Java Swing;
  - or a console (i.e., text) I/O.

Program terminology

- ***Code***: a program or a part of a program.
- ***Source code*** (or *source program*): a program written in a high-level language such as Java.
  - The input to the compiler program, e.g., Java code.
- ***Object code***: the translated low-level program.
  - The output from the compiler program, e.g., Java byte-code.
  - In the case of Java byte-code, the input to the Java Virtual Machine.

Syntax and Semantics

- ***Syntax***: the arrangement of words and punctuations that are legal in a language, the ***grammar rules*** of a language.
- ***Semantics***: the **meaning** of things written while following the syntax rules of a language.

Tip: Error Checking

- ***Bug***: a mistake in a program.
  - The process of eliminating bugs is called ***debugging.***
- ***Syntax error****:* a grammatical mistake in a program.
  - The compiler (or Eclipse) can detect these errors, and will output an error message saying what it thinks the error is, and where it thinks the error is.

Tip: Error Checking

- ***Run-time error:*** an error that is not detected until a program is run.
  - The compiler cannot detect these errors: an error message is not generated during compilation, but during execution (usually the program then dies!)
- ***Logic error:*** a mistake in the underlying algorithm for a program.
  - *The compiler cannot detect these errors, and no error message is generated during compilation or during execution, but the program does not do what it is supposed to do.*

Summary

- Java language
- Java compile model
- JVM and Garbage collection
- Java program environment
- First Java program analysis