This lesson completes the discussion of data abstraction and object-oriented programming with presentations of Java, multiple inheritance, and exceptions.

- **Java** is a popular object-oriented language that has much support for practical programmers.
- **Multiple inheritance** is when a class inherits from more than one class.
- **Exceptions** are an important concept in imperative languages for handling error conditions (both program errors and environment errors).
Introduction to Java

- Java is the most-used language in the world today
  - Supported by libraries, tools, a high-quality implementation (the JVM) and a large developer community
  - But Java is >20 years old: there are many competitors, of which C++, Scala, and Erlang exemplify other parts of the language space
    - C++: closer to the processor architecture; older than Java
    - Scala: a more modern functional/object language built on the JVM
    - Erlang: a multi-agent language for highly available applications
- It is important to understand the execution of Java
  - Examples of Java semantics with the abstract machine
  - Java’s support for object-oriented programming
  - Limitations of Java
Two philosophies: Java versus C++

- Both Java and C++ implement an imperative paradigm supplemented with concurrency
  - (We will discuss concurrency in the next lesson)
  - **Structured programming**: a program is a set of nested blocks where each block has an entry and exit; there is no “goto” instruction in Java (but there is in C++)
  - **Imperative control**: if, switch, while, for, break, return, etc.

- Basic difference in design philosophy
  - C++ allows access to internal representation of data structures; memory management is manual
  - Java hides the internal representation; memory management is automatic (“garbage collection”)
Example program in Java

class Fibonacci {
    public static void main(String [] args) {
        int lo=1;
        int hi=1;
        System.out.println(lo);
        while (hi<50) {
            System.out.println(hi);
            hi=lo+hi;
            lo=hi-lo;
        }
    }
}

- All programs have a method main annotated public static void, executed when the program starts
- A Java variable (argument or local variable) is a cell
- Local variables must be initialized before use
- Integers are not objects but ADTs
- The method println is overloaded – there exist many methods with that name and the implementation chooses the right method according to the argument type (this is also called static polymorphism)
public static void main(...)  

- All methods can be given **modifiers**

- The *main* method has the following modifiers:
  - **public**: visible in the whole program (no restrictions)
  - **static**: there is one per class (not one per object)
  - **void**: the method returns no result
    (so it is a procedure, not a function)

- The *main* method has one argument
  - **String[]**: the argument’s type, an array that contains
    String objects