Java vs C++

CSCI-4448 - Boese

University of Colorado **Boulder**

JAVA vs C++

Java

- Interpreted
- *Write once, run anywhere (platform-independent)*
- The biggest potential stumbling block is speed
  - Interpreted Java runs in the range of 5-20 times slower than C.
  - *But: nothing prevents the Java language from being compiled and there are just-in-time (JIT) compilers that offer significant speed-ups.*

JAVA vs C++

- No separate HEADER-files defining class-properties
  - You can define elements only within a class.
  - All method definitions are also defined in the body of the class. Thus, in C++ it would look like all the functions are inlined (but they're not).
  - Filename and class name must be identical in JAVA
  - Instead of C++ `#include` you use the `import` keyword.
    - *For example:* `import java.awt.*;`
    - *#include does not directly map to import, but it has a similar feel to it*

Java vs C++

- Java
  - Does not have pointers
    - *Avoid unauthorized access of memory locations*
    - *DOES have references…*
  - Does not include struct, union
  - Does not support operator overloading
  - Does not allow preprocessor directives
  - Does not allow global variables
    - *Every method and variable is defined within a class*
  - Does not support multiple inheritance
  - All objects are passed by reference

Java vs. C++

- Java supports
  - Multi-threading
    - *Concurrency*
  - Garbage collection
  - Uses unicode
  - Interfaces
  - Boolean values are predefined literals
    - *C++ false is zero, true is anything that is not zero*
  - Instead of destructor, Java uses finalize()
  - Exceptions are similar, but some in Java are required to be caught.

JAVA vs C++

- Access specifiers (public, private, protected)
  - C++: Controls blocks of declarations
  - Java: Placed on each definition for each member of a class (method and/or variables/constants)
  - Java: Without an explicit access specifier it is the [default], an element defaults to "friendly," which means that it is accessible to other elements in the same package
  - The class, and each method within the class, has an access specifier to determine whether it is visible outside the file.

Java vs C++

- Everything must be in a class.
  - There are no global functions or global data.
    If you want the equivalent of globals, make **`static`** methods and **`static`** data within a class.
  - There are no structs or unions, only classes.
  - Class definitions are roughly the same form in Java as in C++, but there's no closing semicolon.

Java vs C++

- Java has no preprocessor.
  - If you want to use classes in another library, you use **`import`** and the name of the library.
  - There are no preprocessor-like macros.
  - There is no conditional compiling (#ifdef)

JAVA vs C++

- All the primitive types in Java have specified sizes that are machine independent for portability.
  - On some systems this leads to non-optimized performance
  - The char type uses the international 16-bit Unicode character set, so it can automatically represent most national characters.

- Type-checking and type requirements are much tighter in Java.
  - For example:
    - *1. Conditional expressions can be only boolean, not integral.*
    - *2. The result of an expression like X + Y must be used; you can't just say "X + Y" for the side effect.*

Assignment (=) and equality comparison (==)

Assignment (=) and equality comparison (==) have minor differences.

- On primitive (simple) types, = and == are the same in C++ and Java.
- In Java, = and == on classes (or arrays) are comparing references ("pointers"),
- and you cannot overload (redefine) = and == in Java.

JAVA vs C++

- There are Strings in JAVA
  - Represented by the **`String`** class, not renamed pointers
  - Static quoted strings are automatically converted into String objects.

    ```
    String name = "Liz Boese";
    ```

  - There is no independent static character array string like there is in C and C++.

JAVA vs C++

- There are no Java pointers in the sense of C and C++
  - There‘s nothing more to say, except that it is a bit confusing, that a pointerless language like JAVA has a ‘null-pointer‘ error-message... ☺

Parameter Types

C++: a choice of parameter types.

Java: no choice of parameter types

- C++: Call-by-value
  - `void f(int n);`
- C++: Call-by-reference
  - `void f(int& n);`
- Other C++ variants:
  - `void f(const int& n);`
  - `void f(const int n);`

Java: no choice of parameter types

```
public void change(int n)
{
    n = 42;
}
```

This does not change its `int` argument.

There is no way to write a Java method that has a parameter for an `int` variable and that changes the value of an argument variable. Options:

```
int n = computeNewValue();
```

OR use class objects.

```
public class Stuff
{
    private int n;
      ....
    public void changeTheN(Stuff s)
    {
       s.n = 42;
    }
}
```

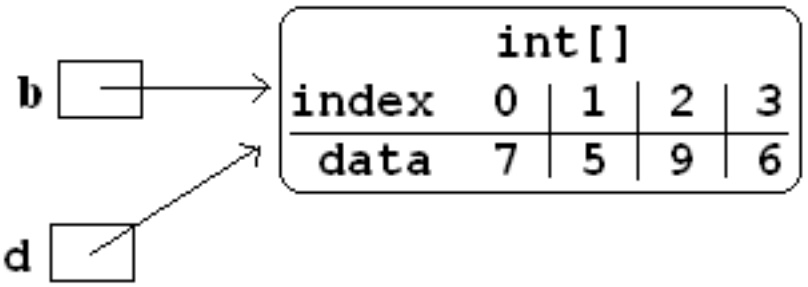
JAVA vs C++

- Arrays: look similar, but have a very different structure and behavior in Java than they do in C++.
  - There's a read-only length member that tells you how big the array is

    `myArray.length` // notice no parenthesis either!
  - Run-time checking throws an exception if you go out of bounds.
  - Can assign one array to another
    - *The array handle is simply copied as aliases*
    - *Shallow-copy!*

```
int[] b = {7, 5, 9, 6}

int[]  d = b
```



JAVA vs C++

- There is a garbage collection in JAVA
  - Garbage collection means memory leaks are much harder to cause in Java, but not impossible.
    - *(If you make native method calls that allocate storage, these are typically not tracked by the garbage collector.)*
  - The garbage collector is a huge improvement over C++, and makes a lot of programming problems simply vanish.
    It might make Java unsuitable for solving a small subset of problems that cannot tolerate a garbage collector, but the advantage of a garbage collector seems to greatly outweigh this potential drawback.

JAVA vs C++

- There are no destructors in Java.
  - There's no need because of garbage collection.

JAVA vs C++

- Java uses a singly-rooted hierarchy, so all objects are ultimately inherited from the root class **`Object`**.
  - The inheritance of properties of different classes is handled by interfaces.
  - Java provides the interface keyword, which creates the equivalent of an abstract base class filled with abstract methods and with no data members. This makes a clear distinction between something designed to be just an interface and an extension of existing functionality via the extends keyword.
  - It's worth noting that the abstract keyword produces a similar effect in that you can't create an object of that class.

JAVA vs C++

- Java has both kinds of comments like C++ does.
- There is no **`goto`** in Java.
  - The one unconditional jump mechanism is the `break` label or `continue` label, which is used to jump out of the middle of multiply-nested loops.
- Java has built-in support for comment documentation
  - **javadoc**: Source code file can also contain its own documentation, which is stripped out and reformatted into HTML via a separate program. This is a boon for documentation maintenance and use.

JAVA vs C++

- Java contains standard libraries for GUIs
  - Simple, robust and effective way of creating user-interfaces
  - Graphical output as part of the language

JAVA vs C++

- Java contains standard libraries for solving specific tasks. C++ relies on non-standard third-party libraries.
  - These tasks include:
    - *Networking, Database Connection (via JDBC)*
    - *Distributed Objects (via RMI and CORBA)*
    - *Compression, Commerce*
    - *Whatever you want: VoIP, Video-Telephony, MIDI, Games,...*
    - *The availability and standard nature of these libraries allow for more rapid application development.*

JAVA vs C++

- Generally, Java is more robust, via:
  - Object handles initialized to null (a keyword).Handles are always checked and exceptions are thrown for failures
  - All array accesses are checked for bounds violations
  - Automatic garbage collection prevents memory leaks
  - Clean, relatively fool-proof exception handling
  - Simple language support for multithreading
  - Bytecode verification of network applets
  - Standard GUI

JAVA vs C++

- Have a look at

http://java.sun.com

- Providing:
- JDK(Java Development Kit)
- Tutorials
- Documentation
- Examples
- APIs (Application Programming Interfaces)
- ...

JAVA vs C++

- ...and it's ***FREE !***