Offline Tutorial : Conversion from C++ to Java

Kevin Wang, kevinw@cse.ust.hk

August 2017

Why are we even reading this.

Throughout the course COMP3111 we are working on a chatbot (chatting-robot) on the instant messager software LINE. One of the easiest way to do is using the example provided by LINE developer site and it is written in Java Enterprise Edition (J2EE). In order to complete the project you will need to pick up a set of skills, which is already *minimized* when we design the project, as listed below:

1. git, *a version controlling repository tools.*
2. Java, *an object oriented programming language.*
3. JUnit, *a testing suite for Java.*
4. Database, *a program that manage your data.*
5. Spring, *a framework that provide web service in Java Enterprise Edition.*
6. Gradle, *a package management tools, like makefile, Ant, Maven.*

As you can see we said "minimized" but there are still a lot to cover. It is impossible to cover all these contents in the lecture or tutorial. Besides, you should also develop the skill to pick up new contents therefore we are here. This document is one of the seven topics tutorial and you are supposed to read this at home and you will be given chance to practise them at lab. In any case you have encounted difficulty in reading this document, you can post your question on the forum or contact the TA for help.

Contents

1 Background

Unless you are an exchange student you must have taken COMP2011/2012 at HKUST already. We assume you are familar with C++ syntax. Some of you may have background in Java, either from a CSE course COMP1022P/COMP1029J or somewhere else. In that case you can skip some chapters while reading this document. This document only provides necessary difference you may need in preparing the project. You will need to reference to other materials like Java Doc or other web resource if you want to learn more about Java. Afterall Java is another programming language and it takes years to familiar with it.

If you are really an exchange student who have no C++ background, or you have returned all your C++ to your instructor after a happy-brain-washing summer, I would suggest you borrow a Java book from the library and read it.

We are not the first one who want to provide a conversion tutorial for C++ programmers to Java. There are some resources you can look at as well.

✍ **Learn More:**

- Learning a New Programming Language: Java for C++ Programmers from University of Wisconsin-Madison *It takes you 5 minutes only.*
- Head First Java *Recommended by many developers but not avaliable at UST library*
- Java programming 24-hour trainer *Can be accessed via UST library*
- Java programming for C/C++ developers by Scott Stricker

2 Hello World and Compile

Let's look at the Hello World program below:

```
//HelloWorld.java  ,  Java source file has an extension .java.
//Most of the time the filename is same as the class name, case sensitive
public class HelloWorld {
    public static void main() {
        for (int i = 1; i <= 10; i++)
            System.out.println("Hello World"+ i);
    }
}
```

Unlike C++, all variables and functions must defined inside a class, even the main function. There is no class header file (.h), i.e., all implementation of the class is written inside the class declaration. Besides, you don't put a semi-color after a class declaration.

To compile this in command line, type:

```
javac HelloWorld.java
```

Then you will generate a classfile called `HelloWorld.class`. It contains something call the byte code that can be executed by the Java-Virtual-Machine (JVM). This byte code theoritically can be executed by the java program on different platform without recompile. To run it, type:

```
java HelloWord
```

There is also an Eclipse for Java as well. In this course we will use a variant of eclipse called Eclipse STS (STS stands for Spring Tools Suites) when developing the chatbot. Before running into the IDE, you may try edit it using any simple text editor like notepad and then compile it with command line. If your computer does not come with a Java Developer Kits (JDK), please install one. You will need Java 8 or above to support lambda expression.

3 Basic Syntax

3.1 Types and Loops

Most of the C++ primitive types like `int`, `double`, etc. can be found in Java. Standard for-loop, while-loop, do-while are used in Java as well.

3.2 Pointers and Reference

Cheers, we don't have pointers and reference symbol anymore. All primitive type are defined as value. All objects and arrays are pointers. Consider the following C++ program

```
List L;          // L is a List, static object;
List *p;         // p is an uninitialized pointer of List;
p = new List;    // now storage for a List has been allocated
                 // and the constructor function has been called
L.AddToEnd(...)  // call L's AddToEnd function
p->AddToEnd(...) // pointer version of calling function
int array[10];   //a static array;
int* arrPtr;       //an integer pointer;
arrPtr = new int[10]; //dynamic array;
List* arrList = new List[10]; // an dynamic array of List.
```

This is how we write it in Java

```
List L;         // L is an uninitialized pointer of List;
L = new List(); // now storage for a List has been allocated
                // and the constructor function has been called;
        // note that you must use parentheses even when you are not
        // passing any arguments to the constructor function
```

```
L.AddToEnd(...) // no -> operator in Java -- just use .
int wrongArray[10]; //compilation error!
int [] array = new int[10]; // dynamic array;
List* arrList = new List[10]; // declaring an array of pointers only
for (int i = 0; i < arrList.length; i++)
 ↪  //arrList.length gives you the size of array!
    arrList[i] = new List();
```

You may also notice the default constructor of a class has a bracket () coming after it. This is not the same as C++. And, no more pointer symbol! Every object (except static object) must be "new-ed" or constructed by other methods when we call it. As there is no pointer syntax anymore, you access the data members or methods of an object with "." symbol directly. [1] This object will be cleaned up by a *Garbage Collector* when it is out-of-scope and no other object referring it, so you don't `delete`.

When it comes to parameters, they are all passed by value. Yet, remember that objects and arrays are pointers, so look at the example from Stack Overflow

```
public static void main( String[] args ) {
    Dog aDog = new Dog("Max");
    // we pass the object to foo
    foo(aDog);

     ↪  // aDog variable is still pointing to the "Max" dog when foo(...) returns
    aDog.getName().equals("Max"); // true, java passes by value
    aDog.getName().equals("Fifi"); // false
}

public static void foo(Dog d) {
    d.getName().equals("Max"); // true
    // change d inside of foo() to point to a new Dog instance "Fifi"
    d = new Dog("Fifi");
    d.getName().equals("Fifi"); // true
}
```

3.3 String manipulation

Let's look at the following C++ program:

```
#include <iostream>
using namespace std;
int main() {
    char* x = new char[10];
    strcpy (x, "abc");
    strcpy (x, "def");
    cout << x;
```

[1] Java does have defined the symbol $->$, it is used in lambda expression.

```
    //ooops memory leak
}
```

This is how we do it in Java:

```
public class StringCopy {
    public static void main() {
        String x = new String();
        x = "abc" + "def";
        System.out.println(x);
        //object x is clean-up automatically here
    }
}
```

Java is relatively friendly to String manipulation. Java does not provide/allow operator overloading except for +, += symbols are defined for String class. Every object in java (I mean every) has a member function (a.k.a method in Java) `toString()` which express the object in a String.

The String class provide many user friendly methods. We table some of them below.

Table 1: Some common methods in String class.

methods	usage	meaning
`equals`	s1.equals(s2)	return true if s1 == s2
+	s1 + 5	concatentate string with integer 5
`length()`		return the length of the string
`substring()`	s1.substring(2, 5)	return a substring of s1 from position 2 to 5
`toLowerCase()`	s1.toLowerCase()	convert s1 to lower case

✍ **Learn More:**

- String API

4 Quick Comparison between C++ and Java

The following table summarizes the C++ that appears our COMP2011 and 2012 but not in Java.

Table 2: Table of "We don't have this"

Syntax	Remarks
* (pointers/dereference)	There are pointer concepts but we never use * any more.
& (reference)	Nope. Parameter is always pass-by-value or pass-by-pointers (for object and array).
`delete`	Memory are auto cleaned up by garbage collector.
`const` variable	We use the keyword final instead.
virtual function	All function are default virtual
protected/private inheritance	But we still have protected/private data members/methods
header file	Just write inside .java directly
`#ifndef #endif`	Pre-processor in Java is not a standard.
const member function	Nope. Annotation ReadOnly introduced in Java 8 does similar things.
const return type	Nope.
destructor	A comparable method is called `finalize()` but you would not know when this will be called. See this
multiple inheritance	Not allowed. See `interface`.

5 Class and Interface

5.1 Constructor

Every class, include self-defined class, in Java is extending directly or indirectly from a primitive class called Object. Same as C++ if you don't write a constructor a default constructor will be assigned. To copy an object we use the method `clone()` which is inherit from the class Object, instead of calling a copy constructor.

5.2 Public/Protected/Default/Private Class

To make things easy, follow these rules:

1. One file one class;
2. Make the class public;
3. Name the filename same as the class name (case sensitive).

It gives you a working solution but it may not be a good design. So there are different type of class for the different levels of encapsulation.

✍ **Learn More:**

- Explanation of the concept on Stack Overflow

5.3 Inheritance

The syntax for inheritance in Java is the keyword `extends` (with a 's') such as

```
class A extends B {
    public A() {  //constructor
         super("Name");
         System.out.println(name);
    }
}

class B {
    protected String name;
    public B(String n) {
        name = n;
    }
}
```

There is no private/protected inheritance in Java. There is no member initialization list (MIL) in Java as well. To choose an appropriate constructor from your parent, call the appropriate parent's constructor `super(<some parameter>);` in the first line of your constructor.

5.4 Virtual Function

All non-static methods are by default virtual. Thus, virtual function not a common term in Java. If you don't want a method not to be overrided, declare it as `final`. When you override a non-abstract method, you are advised to put an *annotation*, `@Override`, right before the function. We will explain about annotation in the later section. For now, just consider annotation is nothing but a hash-tag.

```
class A  {
    public void test() {
        System.out.println("Class A");
    }
}

class B extends A{
    @Override  //this is optional but recommended
    public void test() {
```

```
        System.out.println("Class B");
    }

}
class C extends B{
    public final void test() {
        System.out.println("Class C, mic-dropped.");
    }

}
class DoesNotWork extends C {
    public void test() {} //cause compilation error
}
```

✍ **Learn More:**

- When do you use java's override annotation and why

5.5 Abstract Class

The syntax for abstract class in Java is the keyword `abstract` and an ABC must have a pure-virtual function (a.k.a abstract method in Java). We don't write =0 here, instead, we say abstract.

```
abstract class Student {
    protected int mark;
    abstract void setMark(int x); //no implementation is allowed
}
```

5.6 Interface

You may aware you can inherit two different classes in C++, which is called *multiple inheritance*. This is not allowed in Java. One of the reason is that if two parent classes have the same method it is impossible to resolve which implementation you want to do. In this case how would you define a toy that is also a food?

Interface works like an abstract class but it does not allow any implementation. In C++ terminology, an ABC with only pure virtual function. Commonly known interfaces are `Runnable`, `Serializable`, `Throwable`, etc. The following shows an interface.

```
interface Eatable {
    public int getCalorie();
}
class Kinder extends Toy implements Eatable {
    public int getCalorie() {return 100000;}
```

```
    //rest of the code
}

class Child {
    int weight;
    public void eat(Eatable e) {
        weight += e.getCalorie() / 1000;
    }
}
```

✍ **Learn More:**

- Java Tutorial: Interface
- Discussion on Stack Overflow about Interface and Abstract Class

6 Generic Programming (Template)

The generic programming of Java is very similar to C++ while it does not require you to write the line `template<class T>`.

✍ **Learn More:**

- Oracle Java Tutorial: Generics

7 Try/Catch, Exception handling

Try and Catch is also defined in C++ but we are not dealing with that in our 2011/2012. Just consider that you are required to write a function to read the content a file and return that as an object. The filename is passed as an argument into your function. There are tons of reasons (we call them exceptions) that you will terminate the function, like, incorrect filename, inaccessible disk, permission problem, etc. This is not your function's responsibility to handle all those issues and you want to make sure someone is going to handle these *exceptions*.

A method may `throws` an `Exception`. A caller of this method must either 1) Throw that exception (let the caller of it to handle it); or 2) Catch and handle it. Look at the following code:

```
import java.util.Random;   //loading a library
//You can also write import java.util.*;
public class A {
    public static void main(String argv[]) {
```

```
        A a = new A();
        try {
            a.danger();
        } catch (CustomException ce) {
            ce.debug();
        } catch (ArrayIndexOutOfBoundsException aiobe) {
            System.out.println(aiobe.toString());
        } finally {
            System.out.println("Come here anyway");
        }
    }
    public void danger() throws CustomException,
    ↪ ArrayIndexOutOfBoundsException {
        Random r = new Random();
        System.out.println(1);
        if (r.nextInt() > 100)
                throw new CustomException();
        else
                throw new
                ↪ ArrayIndexOutOfBoundsException("You are bad luck today");
    }
}

class CustomException extends Exception {
    public void debug() {
        System.out.println("Hello, this is a debug message");
    }
}
```

The method `danger()` may throw two different types of Exceptions. The `main` function is required to enclose the method with a `try-catch` block or otherwise it wouldn't compile. There are many default Exceptions and you can define your own Exceptions like the above example.

✍ **Learn More:**

- TutorialPoints Java Exception

8 Annotation

Annotation was introduced in Java JDK version 1.5, in 2002 as to "tag" a program. The tag can be set to be read by human programmers, the compiler, or/and the JVM. There are many usage of annotation and people love using it, as much as they like to use hash tags in the social media.

An annotation always starts with the symbol `@`, mostly before the declaration of a class, a method, or a variable. But it can also be placed when a type is used from Java 8 onwards. Consider the following code in inheritance, with the annotation `@Override` the compile will generate an error per your typo.

```
public class Child extends Parent {
    @Override
    public void tests() {}


}

class Parent {
    public void test() {} //without s
}
```

✍ **Learn More:**

- Java Tutorial: Annotation

9 Self-Test

1. What is the filename extension for a Java source file?
2. What command can compile a Java file?
3. How to declare an array of String?
4. How to declare a 2D array of int?
5. How to compare two Strings are equal?
6. How to declare an ABC?
7. What is the main different between an ABC and an interface?
8. What is Try-Catch? What is the alternative if I don't write Try-Catch in my function when I need to?
9. Do I ever click the links in learn more?