VE572 — Methods and Tools for Big Data

Lab 1

TA: Yihao Liu — UM-JI (Summer 2019)

Goals of the lab

- Basic file input / output in Java
- Object-oriented programming in Java

1 Introduction

This lab requires you a strong background of C++ programming and helps you to turn it into the ability of Java programming. A simple guide can be found in the appendix cpp2java.
...

2 Appendix

VE572 — Methods and tools for big data

From C++ to Java

Jing and Manuel — UM-JI (Summer 2019)

Content of this document

- No information on syntax
- Introduction to major Java concepts
- Hints on important topics to understand

A C++ Hitchhiker's Guide to the Javanian Galaxy

Java is a compiled language, yet it runs on the Java Virtual Machine (JVM).
Java programs need to be compiled and run on a runtime called Java Runtime Environment (JRE). The Java development environment requires a package called Java Development Kit (JDK).

Java is fully object-oriented.
Java programs are organized in terms of classes. All variables or functions must reside within the scope of a class. All classes always inherit a common base class "Object".

The Java memory management features a garbage collector.
Object are automatically deleted and recycled when no longer needed. However, in some cases, you have to pay a significant performance price for that feature. Stay alert.

Java is statically and weakly typed.
Like C++, all Java variables are typed, and their type must be known at compile time. If necessary variables can be cast.

Java objects are always passed by reference.
Java contains a few built-in types that are passed by value. However all objects are passed by value-references. Think of them as pointers that automatically dereferences themselves.

Java is polymorphic by default.
By default, all methods in a class are "virtual". Subclasses always override base class objects.

Java allows only single Inheritance and features Interfaces.
Only single inheritances are allowed. Interfaces in Java are similar to pure abstract base classes in C++.

Generics are done by type-erasure.
Java also supports Generic containers, similar to `std::vector<>` or `std::list<>` in C++. However, unlike C++, Java uses type-erasure to handle generics. In contrast, C++ uses type specialization.

Java supports reflection and introspection.
It is possible to ask a class to print out its supported methods, or look up a class by string, at runtime.

Have fun with Java.
Yep. Enjoy Coding!