CSE 1325 - Object-Oriented Programming

Introduction

Alex Dillhoff

University of Texas at Arlington

Course Description

Object-oriented concepts, class diagrams, collection classes, generics, polymorphism, and reusability. Projects involve extensive programming and include graphical user interfaces and multithreading.

Object-Oriented Programming

The official course description makes no mention of the specific language used.

The focus of this course will be on learning the fundamentals of object-oriented programming in a language-agnostic way.

Object-Oriented Programming

What is object-oriented programming?

In **procedural programming**, programs are created by breaking down problems into modular components designed to solve specific subtasks.

Interfacing with the data is also facilitated through separate libraries and functions.

Object-Oriented Programming

Object-oriented programming breaks down a solution into objects which **encapsulate** some concept.

This is often viewed as a data-driven approach, in which the objects and actions are defined based on the data.

Object-Oriented Programming

Procedural programming works well for small to medium-sized projects.

As the complexity and scope of a project grows, it can be difficult to keep the code organized.

This is not the case for object-oriented programming. Since variables and functions are defined based on specific concepts, they are easier to organize.

Object-Oriented Programming

There are many languages which include object-oriented programming concepts.

The specific language we will use in this course is **Java**.

Java

The original authors described their design decisions and goals for the language in a white paper.

The nuances of designing a programming language will be discussed in a future class, but it is important to review some of the reasoning which led to the creation of Java.

Simplicity

The authors looked at the complexity, at the time, of languages such as C++.

This informed their design to include native features that supported modern programming practices.

Object-Oriented

Java supports object-oriented programming techniques.

These concepts put the focus on the data and interfaces to the data.

Robust

Java provides interfaces for checking and recovering from run-time errors.

These features come on top of the usual compile-time checks that a compiler would provide.

Security

Java is executed through a virtual machine that essentially runs each program through an isolated "sandbox".

Additionally, a code signing standard is enacted to digitally sign code that is known to be trusted.

Architecture-Neutral

Code written in Java compiles to an architecture-netural format that can further be translated into the target machine code.

This allows any Java code to be executed on any system that supports the Java runtime system.

Portable

In contrast to C and C++, Java does not have any implementation-dependent features.

That is, data types and their operators behave the same regardless of the underlying implementation.

Interpreted

Java even provides a Read-Eval-Print Loop (REPL).

This allows code to be interpreted and executed on the fly.

High-Performance

Through the use of Just-In-Time (JIT) compilation, frequently executed code can be optimized to reduce the runtime of an application.

Many advances to JIT compilers have been made over the years. Today, they are competitive with traditional compilers.

Getting Setup for Java Development

- Install the JDK
- Learn the command-line tools
- (Optional) Set up an IDE

Installing the JDK

For this class, we will be using the latest version: **Java 20**.

It is available from Oracle's website:
`https://www.oracle.com/java/technologies/downloads/`

Using the Command-Line Tools

To ensure that the setup was successful, we will learn how to use the tools to compile and run Java programs.

You may use an IDE in this class. However, you **must** know how to compile and run programs from the command line.

Compiling Java Programs

First, let's write a simple program.

```
public class FirstProgram {
    public static void main(String[] args) {
        String msg = "Welcome to CSE1325!";
        System.out.println(msg);

        for (int i = 0; i < msg.length(); i++) {
            System.out.print("=");
        }

        System.out.println();
    }
}
```

Compiling Java Programs

Save the program as `FirstProgram.java`.

By convention, you should name your code files to match the name of the **class**.

Compiling Java Programs

To compile the program, use the tool `javac`.

For example, the previous program can be compiled by typing

```
javac FirstProgram.java
```

This generates a file `FirstProgram.class`.

Running Java Programs

At this point, the program can be run via

```
java FirstProgram
```

Note that the postfix `.class` does not need to be included.

Running Java Programs

Starting with JDK version 11, programs are contained within a single class file.

That is, the program can be run directly via

```
java FirstProgram.java
```

JShell

Java can be interpreted via REPL.

Code can be evaluated using `JShell`

JShell

To start `Jshell`, simply type `jshell` at the prompt.

```
$ jshell
```

You can execute Java statements individually.

JShell

JShell Example

Using an IDE

You may use an IDE for this class.

Keep in mind that you should only submit the relevant code files for each assignment.

Using an IDE

There are several useful IDEs available for Java.

I recommend **Visual Studio Code** available at
`https://code.visualstudio.com/`

Using an IDE

IDE Setup