Intro to Java & Eclipse

References

- tutorialspoint: Java Tutorial
- Oracle Java doc:
    - Getting Started
    - Language Basics
    - Classes & Objects

Hello, World!

1. Create and run a "Hello, World!" project in Eclipse:
    a. Start Eclipse
    b. Choose a workspace (e.g. S:\CP2\workspace)
    c. Go to the workbench
    d. Create a new Java project **HelloWorldApp**
    e. Create a new class **HelloWorldApp:**

```
/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 */
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
```

Java Basics

- Syntax vs. Python / C++
    - Java syntax is very similar to C++
    - Wikipedia: Comparison of Java and C++
- Java is object-oriented from the ground up, even your "Hello, World!" program needs to implement a class!
- Java class terminology:
    - Class data members are called **fields**.
    - Class functions are called **methods.**
- In Java, every application must contain a main method with this **signature (header)**:

- `public static void main(String[] args)`

Example class

Person class template

- This is a template only - notice that the methods are not implemented.
- create a new class Person, or new file Person.java and copy the code below

```
public class Person {

	/************ Fields - default access level **********/

	String _fName;	// first name
	String _lName;	// last name
	String _DOB;	// date of birth in "MM-DD-YYYY" format

	/************* Constructors **************/

	// initialize fields to default values
	public Person(){ }

	// initialize fields using parameters
	public Person(String f, String l){ }

	// initialize fields using parameters
	public Person(String f, String l, String dob){}

	/************* Overrides **************/

	// return a String that represents this object
	public String toString(){ return "not implemented"; }


	/************* Private method(s) ***********/

	// return true if dob is in valid "MM-DD-YYYY" format, else false
	private boolean validDOB( String dob ){ return true; }

	/************* Getters and setters **************/

	// return value of _name
	public String getName(){ return "not implemented"; }

	// change value of _name
	public void setName(String first, String last){ }

	/************* Main **************/

	// construct Person objects and test methods
	public static void main(String[] args){
		Person p1 = new Person();
		System.out.println( p1 );

		p1.setName( "new", "name" );
		// etc...
	}
}
```

Examining the Person class

- **Constructors**
  - The **default constructor** is the no-parameter constructor - Person()
  - **Overloading** is the process of creating multiple methods with the same name but different parameters.
  - This allows you to use a method in different ways - see the 3 versions of the Person object constructor below.
- **toString**
  - **toString** is a method of Object that returns a String representing the object.
  - toString is called implicitly (automatically) when you print an object using System.out.println.
  - You should define toString in (just about) every class you create.
- **Access modifiers**
  - From least to most restrictive: { public, protected, (default), private }
  - Right now we'll just use default (no modifier) or public.
    - **default** - any class in the same package can access.
    - **public** - any class in the same project can access.
- **Getters and setters**
  - In many situations, another class without access to a field will need to get or set the field's values. To support this, you implement methods to **get / set.**
  - It's good practice to use getters and setters, so that you can change the underlying **implementation** of a class without changing the **interface.**
- **main**
  - main is a **static method**
  - **static methods (or fields)** are a.k.a. class methods - they exist independently of any created objects
  - It's best for now to just think of main in C++ terms - as if it is a separate entity from your class.