**Computer Science CSC 324S: Programming Languages**
**University of Toronto – St. George Campus**

# Assignment 1A.a, Intro Java

**Due Date: Tuesday, 26 January, 6:10pm exactly**

## Administration

Two thing:

- This is the first part of assignment 1A. There will be another part, handed out next week. This is distinct from assignment 1B, the second Java assignment. If you are worried about the size of these, you had better start now so you can have some influence over exactly how large the next part will be. If I don't hear from you, I'm assuming that the workload is just right. `:-)`
- Next week, the second part will include a description of what to hand in, how to submit files in CDF, and exactly how to hand in your paper version.

## Introduction

You've just been hired by the clothing store The Gap to create a prototype of a product inventory system. The previous programmer learned just enough to cobble together a driver program, but all it does is read in a character, and if it's a `q`, then the program quits. Since this took four months to do, that programmer has been fired.

## Programming Style

Here are the minimum style requirements, as laid out by your supervisor:

- You must indent your code — at least two spaces for each indentation level. (Although I find four more readable.)
- You must comment your code clearly. Every variable must have a comment stating what it is used for. Every method must have a comment describing its parameters and what the method does. For public methods, the comment must not reveal any implementation details. Every class must have a comment describing what it represents.
- In all your code, every class and interface must start with a capital letter. Every one of them. You will lose 1 mark for every non-capitalized class and interface. Example of a good class name: `class Freeble`. Example of a bad class name: `class breeble`.
- In all your code, every item that is not a class (such as packages, methods, and parameters) must start with a lowercase letter. You will lose 1 mark for every capitalized non-class/interface. Example of a good variable name: `moogah`. Example of a bad variable name: `Hooey`.
- In all your code, please capitalize every internal word. Examples: `aLongVariableName`, `ALongClassName`. We will be more flexible on this requirement, but please try to adhere to it.

## Part A

Design a class hierarchy for clothes. Include information like price range (in dollars), size (measured in various ways), fabric (one of cotton, wool, polyester, and acrylic – no blends), and colour (one of persimmon, blue, white, black, brown, paisley, and plaid). Include appropriate comments for any instance variables and methods that you include. Include any appropriate abstract classes and interfaces.

**Hand in:** a printout of the class hierarchy. Use the format found on the following web page:

`<http://java.sun.com/products/jdk/1.1/docs/api/tree.html>`

Include package names. As stated previous, you must also include instance variables and methods – indent them as well, and indicate their type and visibility modifier. Only show the instance variables and methods in the classes that define them – don't include them in subclasses.

Here are the various products; you will have to decide on a sensible hierarchy. (Luckily, the Gap you work for only sells men's clothing, and has a somewhat strange sizing system.) (In other words, I'm clueless about women's styles.) Feel free to make up new products.

- Products (price range in parentheses: t-shirts ($5-$15), casual shirts ($10-$50), dress shirts ($30-$60), sweaters ($40-$120), dress pants ($40-120$), jeans ($30-$50), shorts ($15-$60).
- T-shirts, casual shirts, and sweaters come in three sizes: small, medium, and large.
- Dress shirts are arranged by collar size (range 13-21).
- Dress pants and jeans are sold by the combination of two measures: waist size (range 28-44) and inseam length (range 28-40).
- Shorts are sold by waist size only (range 28-44).
- Dress pants must not be polyester.
- Jeans must be cotton and blue.
- dress shirts must not be persimmon, plaid, or paisley.

# Part B

Implement your hierarchy.

# Part C

Modify the driver program (available on the web page) to maintain an inventory list that allows the following operations:

- Add an item to the inventory list
- Sell an item from the inventory list
- Find out how many items there are
- Find out how many of a particular item there are
- Find out how many of the various intermediate classes there are
- Find out how much money has been made

**Hint:** Read about the `instanceof` keyword in the Java Glossary, linked from the course web page. Also read up on the method `Class.forName()` and `Class.isInstance()`. There are several ways to test the types of `Objects`; any one of them will do. Note that some require you to deal with `Exceptions`; here is some code that will take care of it:

```
try {
    Driver d = new Driver();
    Class blerg = Class.forName("csc324.driver.Driver");
    if ( blerg.isInstance(d)  ) {
        System.out.println("Whee!");
    }
} catch (Exception e) {
    System.out.println("Exception happened.");
    e.printStackTrace();
}
```

Note that some operations will need more information (such as the size, colour and price of an item to be created or deleted). It is up to you to decide how to do that.

It will also help you tremendously to create an `Inventory` class that somehow stores all the items. It is up to you to decide how to store them. However, there must not be a maximum number of items. **Hint:** look up class `Vector` in the Java API.