Managing Objects

Fall 2015

Computer Science
UNIVERSITY OF TORONTO

StudentManager

Responsibilities:

- Reading `Student` objects from a file to a `Map`.
- Maintaining a `Map` of student id to `Student` objects when the app is running.
- Writing `Student` objects from a `Map` to a file.

First launch of the application

`StudentManager` is responsible for:

- Reading data from a CSV file.
- Constructing `Student` objects based on data from the CSV file and populating a `Map` with those `Student` objects.

Before the application terminates

`StudentManager` is responsible for:

- Writing `Student` data to file.

Which file format should be used?

If a CSV file is used, then the next time the application is launched, we would need to parse the file and reconstruct the student objects by calling on the `Student` constructor and passing in arguments.

Persistent data

Sometimes our code computes something and when the program stops, we’re done. Each run of the program is a fresh start.

Other times, we need the results of the previous run to feed in to the next run of the program.

How do we make that data persist between runs of the program?

What does the code look like?

Serializable Data

Rather than writing the values of an object's instance variables to file, a representation of the object itself can be written to file.

An object is *serializable* if it can be be represented as a sequence of bytes.

The *serialized* object can be written to file.

The object can later be *deserialized*. That is, the object can be reconstructed using the data read from the file.

Java’s interface `Serializable`

Java provides interface `Serializable` to serialize objects.

In order for a class to be serializable, it and its ancestor(s) must implement the `Serializable` interface and every instance variable in the class must also be `Serializable`.

All of Java’s primitive types are serializable. For class types, check the Java API.

Obligations imposed by `Serializable`

There are none!

In other words, for your class to be `Serializable` you simply need to say `implements Serializable`.

How can Java write/read instances of *any* class?

What could the write/read method look like??

`StudentManager`

Responsibilities (revised):

- For the first launch of the application, reading `Student` information from a CSV file, constructing `Student` objects, and populating a `Map`.
- Reading *serialized* `Student` objects from a file to a `Map`.
- Maintaining a `Map` of student id to `Student` objects when the app is running.
- Writing *serialized* `Student` objects from a `Map` to a file.