In this workshop, you store a set of polymorphic objects in an STL container.

**LEARNING OUTCOMES**

Upon successful completion of this workshop, you will have demonstrated the abilities to

* store and manage polymorphic objects using an STL vector
* store a set of uniform rates for all instances of a class using a class array
* report and handle an exception

**SPECIFICATIONS**

This workshop consists of three modules:

* **w6** - the main program receives the name of the file that holds information about the products sold to a customer
* **iProduct** hierachy - each object describes a single product sold to the customer
* **Product, TaxableProduct** – concrete base classes

The **main** program receives a user-specified file (Sales.dat) via command line arguments.  The records hold price information about products, some of which are taxable.  Taxable and non-taxable products belong to the same hierarchy; the taxable product class derives from the simpler product class.

File Records

Each record in the file consists of two or three fields: a product number field, a product cost field and an optional taxable code field.  The cost field contains the price before any tax.

10012 34.56

10023 45.67 H

10234 12.32 P

10056 67.54

10029 54.12

10034 96.30

* **H** identifies a taxable product at the HST rate
* **P** identifies a taxable product at the PST rate

A non-taxable record terminates with a newline character immediately following the price field.  A record with a taxable code other than the two listed above is an invalid record.  A file with an invalid record is a corrupted file.

iProduct Hierarchy

The **iProduct** hierarchy holds information about the different products purchased by a customer.  The interface to this hierarchy is defined in the iProduct.h file provided.

* The **double getPrice() const** query returns the price chargeable to the customer.
* The **void display(std::ostream& os) const** query receives a reference to the output stream and inserts the product information in the form of a single line into the output stream.
* The overloaded insertion operator (**<<**) receives a reference to the output stream and displays the product information in the form of a single line on the output stream.
* The **readProduct** function receives a reference to the input file stream, reads the next record from the stream, allocates memory for the corresponding **iProduct** object, copies the record information into that object and returns its address to the caller function. If the tax code is not valid (ie. H or P, a **std::string** should be thrown.

Implementations

Complete the definition of the polymorphic **iProduct** object by implementing two distinct classes:

* **Product** - holds information about a product that consists of its product number and its cost
* **TaxableProduct** - derives from **Product** and holds additional information about the product's taxable status

The same tax rates apply to all **TaxableProduct** objects.  Use a class array (an array of class variables) to store the tax rates for HST and PST tax codes (13% and 8%).

Main Program

Complete main program provided by adding a vector of iProduct\* and populating the STL container with the data from the Sales.dat input file.  If an exception is thrown, the main program returns an error code of 2.

The output from your completed main program for the file contents listed above should read like:

Product No Price Taxable

10012 34.56

10023 45.67 HST

10234 12.32 PST

10056 67.54

10029 54.12

10034 96.30

Total 317.43