WICHITA STATE UNIVERSITY

CS 311: Object Oriented Programming
Spring 2018 – Assignment 5

For this assignment, the program for assignment 4 is carried forward and modified to demonstrate the two object-oriented concepts of class inheritance and polymorphism. To accomplish this, the grain sample within a ticket will be identified as one of three grain types: Wheat, Soybean, Milo

Use the following criteria for writing your program:

- ✓ Modify existing class named Grain (header file named Grain.h, source file named Grain.cpp) to:
  - Define/implement a polymorphic class destructor member function
  - Define an abstract (pure-virtual) polymorphic member function named *clone()* to return a pointer to a clone (copy) of the calling object
  - Define an abstract (pure-virtual) polymorphic member function named *getType()* to return a string representing the calling object's grain type
  - Define an abstract (pure-virtual) polymorphic member function named *getAverageTestWeight()* to return the grain's average test weight (lbs/bushel) as a constant
  - Define an abstract (pure-virtual) polymorphic member function named *getIdealMoistureLevel()* to return the grain's ideal moisture level (%) as a constant
  - Define/implement a polymorphic member function named *toString()* to return a string representing the calling object's moisture level and foreign material (ex. "Moisture level: 14.0%, Foreign material: 0.75%")

- ✓ Implement derived class named *Wheat* (header file named Wheat.h, source file named Wheat.cpp) whose base class is *Grain* to:
  - Declare constants (limited to file scope only) for average test weight = 60.0 and ideal moisture level = 13.5
  - Define/implement a default class constructor to initialize each member variable defined in the base class to 0 (hint: use constructor delegation by invoking the base class default constructor within the initialization section)
  - Define/implement an overloaded class constructor to initialize each member variable defined in the base class to the value of its corresponding parameter provided (hint: use constructor delegation by invoking the base class overloaded constructor within the initialization section)
  - Define/implement a member function named *clone()* to return a pointer to a clone (copy) of the calling object
  - Define/implement a member function named *getType()* to return the string "Wheat" representing the calling object's grain type
  - Define/implement a member function named *getAverageTestWeight()* to return the grain's average test weight (lbs/bushel) as a constant (hint: declare a constant and return it)
  - Define/implement a member function named *getIdealMoistureLevel()* to return the grain's ideal moisture level (%) as a constant (hint: declare a constant and return it)
  - Define/implement a member function named *toString()* that overrides the base class *toString()* member function and returns a string inclusive of the calling object's grain type (hint: invoke *getType()* and base class *toString()* member functions)

- ✓ Implement derived class named *Soybean* (header file named Soybean.h, source file named Soybean.cpp) whose base class is *Grain* to:
  - Declare constants (limited to file scope only)for average test weight = 60.0 and ideal moisture level = 13.0

- Define/implement a default class constructor to initialize each member variable defined in the base class to 0 (hint: use constructor delegation by invoking the base class default constructor within the initialization section)
- Define/implement an overloaded class constructor to initialize each member variable defined in the base class to the value of its corresponding parameter provided (hint: use constructor delegation by invoking the base class overloaded constructor within the initialization section)
- Define/implement a member function named *clone()* to return a pointer to a clone (copy) of the calling object
- Define/implement a member function named *getType()* to return the string "Soybean" representing the calling object's grain type
- Define/implement a member function named *getAverageTestWeight()* to return the grain's average test weight (lbs/bushel) as a constant (hint: declare a constant and return it)
- Define/implement a member function named *getIdealMoistureLevel()* to return the grain's ideal moisture level (%) as a constant (hint: declare a constant and return it)
- Define/implement a member function named *toString()* that overrides the base class *toString()* member function and returns a string inclusive of the calling object's grain type (hint: invoke *getType()* and base class *toString()* member functions)

✓ Implement derived class named *Milo* (header file named Milo.h, source file named Milo.cpp) whose base class is *Grain* to:

- Declare constants (limited to file scope only)for average test weight = 56.0 and ideal moisture level = 13.0
- Define/implement a default class constructor to initialize each member variable defined in the base class to 0 (hint: use constructor delegation by invoking the base class default constructor within the initialization section)
- Define/implement an overloaded class constructor to initialize each member variable defined in the base class to the value of its corresponding parameter provided (hint: use constructor delegation by invoking the base class overloaded constructor within the initialization section)
- Define/implement a member function named *clone()* to return a pointer to a clone (copy) of the calling object
- Define/implement a member function named *getType()* to return the string "Grain Sorghum" representing the calling object's grain type
- Define/implement a member function named *getAverageTestWeight()* to return the grain's average test weight (lbs/bushel) as a constant (hint: declare a constant and return it)
- Define/implement a member function named *getIdealMoistureLevel()* to return the grain's ideal moisture level (%) as a constant (hint: declare a constant and return it)
- Define/implement a member function named *toString()* that overrides the base class *toString()* member function and returns a string inclusive of the calling object's grain type (hint: invoke *getType()* and base class *toString()* member functions)

✓ Modify existing class named *Ticket* (header file named Ticket.h, source file named Ticket.cpp) to:

- Modify member variable for grain sample to be a pointer of type *Grain*
- Modify default class constructor to initialize member variable pointer to grain sample using *nullptr* value
- Modify overloaded class constructor to allow caller to pass a pointer to grain sample instead of a reference. Initialize pointer (if not null), using *clone()* member function to create copy (**do not** simply copy value of pointer provided by caller)
- Define/implement a class copy constructor to create a copy of the *Ticket* parameter provided (hint: must also create a copy of grain sample using *clone()* member function (if not null))
- Define/implement a class destructor member function to delete memory used for grain sample (if not null)

- Define/Implement a member function *getSample()* to return a pointer, if it exists, to a new copy of the grain sample of the calling object (hint: create a copy of grain sample using *clone()* member function (if not null) or return *nullptr* value)
- Modify member function *toString() to* return a string inclusive of the grain sample's type (hint: invoke *getType()*)
- Define/implement an overloaded assignment operator = member function to create a copy of the *Ticket* parameter provided (hint: must check for assignment to self, delete memory used for existing grain sample of the calling object (if not null), create a copy of grain sample (if not null))

✓ Modify existing class named *Input* (header file named Input.h, source file named Input.cpp) to:

- Define an enumerated type named *Type* with the set of values *WHEAT*, *SOYBEAN*, *MILO*
- Prompt user for grain sample's type (wheat, soybean, milo (grain sorghum)) and create grain sample objects on the heap based on type (hint: could utilize enumerated type for determining grain type)
- Modify creation of Ticket object to provide pointer to grain sample

To test the changes, use sample input below and validate output. Most of the changes are a result of refactoring, so the user should not see any noticeable differences except the addition of prompting the user for the grain sample's type and seeing the sample's type on the receipt for each ticket.

Example Input:

```
Ticket number 1 (alphanumeric): 101300A
Gross weight (lbs): 33180
Tare weight (lbs): 10780
Moisture level (%): 14.0
Foreign material (%): 0.75
Grain type (w)heat, (s)oybean, (g)rain sorghum: w

Ticket number 2 (alphanumeric): 101400A
Gross weight (lbs): 24150
Tare weight (lbs): 10780
Moisture level (%): 11.5
Foreign material (%): 1.25
Grain type (w)heat, (s)oybean, (g)rain sorghum: g

Ticket number 3 (alphanumeric): 101300A
Gross weight (lbs): 31200
Tare weight (lbs): 10780
Moisture level (%): 13.25
Foreign material (%): 2.0

Error: Duplicate ticket encountered, ticket ignored!

Ticket number 3 (alphanumeric): <nothing entered, user just presses Enter>
```

Example Output (based on input):

```
Wheat Ticket 101300A – 02/13/2017 16:05:00:
    33180   Gross Weight
    10780   Tare Weight
    22400   Net Weight

   373.33   Gross Bushels
     7.47   Moisture Level (14%)
     2.80   Foreign Material (0.75%)
   363.07   Net Bushels

Grain Sorghum Ticket 101400A – 02/13/2017 16:05:30:
    24150   Gross Weight
    10780   Tare Weight
    13370   Net Weight

   222.83   Gross Bushels
     0.00   Moisture Level (11.5%)
     2.79   Foreign Material (1.25%)
   220.05   Net Bushels

Harvest Summary Totals
   596.17   Gross Bushels
   583.11   Net Bushels
```

Requirements and Submission:

- Due date: Saturday, March 17, 11:59pm
- Include the following header information (comment lines) at the beginning of each source/header file (before the #include directives) submitted. Notice each comment line begins with two slashes. C++ recognizes these as the start of single line comment. If you prefer, you may use block commenting.

```
// File Name: Harvest.cpp
// Author: Firstname Lastname
// Student ID: ********
// Assignment Number: ##
```

Such as:

```
// File Name: Harvest.cpp
// Author: Joe Shobe
// Student ID: xxxxxxxx
// Assignment Number: 5
```

- Grades will be based on:
  - ✓ User-friendly interactive input, output, and error messages (when applicable)
  - ✓ Programming style: Proper use of control statements, standard C++ techniques and naming conventions, and the conciseness and readability of the code
  - ✓ Accuracy: Does the program implement required changes, calculate and output correct data
  - ✓ Appropriate and descriptive comments
- Use of solutions found on the web, in the books, or from other students is <u>not</u> permitted. It must be your own work. Under no circumstances are you to share your work with another student.
- Review the section *Submitting Assignments* in the Linux Server PDF for more information on how to turn in your assignment.
- No submissions will be accepted by Blackboard, e-mail, CD, DVD, or other independent storage or electronic media, unless approved by the instructor.