```cpp
/*
 * Program name:  Smuggler (Programming Project 02)
 * Program description: Driver for the smuggler game which handles the creation and management
 *      of SmugglerShip and TradeItem instances.  Provides a simple menu so that the user can
 *      interact with the functions within these objects.
 *
 * Programmer:  Chad Philip Johnson
 * Date created: February 21st, 2013
 * Last date modified: May 10th, 2013
 *
 * Sources Used:
 *      CinReader.h
            - for handling user input and to ensure that valid values are passed into the program
 *      smugglership.h
 *          - for accomodating created instances of the SmugglerShip class
 *      tradeitem.h
 *          - for accomodating created instances of the TradeItem class
 */

#pragma once

#include <cstdlib>
#include <iostream>
#include <string>

#include "smugglership.h"
#include "tradeitem.h"
#include "CinReader.h"

using namespace std;

/* for unit testing -- do not alter */
template <typename X, typename A>
void btassert(A assertion);
void unittest ();

/**
 * Menu system for interactive test.  Allows the user to work with the astros aboard a SmugglerShip object.
 * @param objProcessUserInput Instance of CinReader to handle user input.
 * @param objSelectedSmugglerShip The instance of SmugglerShip for the user to interact with.
 */
void workWithAstros( CinReader &objProcessUserInput, SmugglerShip &objSelectedSmugglerShip);

/**
 * Menu system for interactive test.  Allows the user to work with the trade items aboard a SmugglerShip object and the ship's
 * cargo capacities.
 * @param objProcessUserInput Instance of CinReader to handle user input.
 * @param objSelectedSmugglerShip The instance of SmugglerShip for the user to interact with.
 * @param objSelectedTradeItem The instance of TradeItem for the user to interact with.
 */
```

```
51  void workWithItems( CinReader &objProcessUserInput, SmugglerShip &objSelectedSmugglerShip, TradeItem &objSelectedTradeItem );
52
53  /**
54   * Menu system for interactive test.  Allows the user to work with the properties of the ship.
55   * @param objProcessUserInput Instance of CinReader to handle user input.
56   * @param objSelectedSmugglerShip The instance of SmugglerShip for the user to interact with.
57   */
58  void workWithShip( CinReader &objProcessUserInput, SmugglerShip &objSelectedSmugglerShip );
59
60  int main (int argc, char* argv[])
61  {
62      unittest();
63
64      CinReader objProcessUserInput;
65      char charUserInput      = '0';
66      int intUserInput        = 0;
67      bool boolContinue       = false;
68      SmugglerShip *pobjSelectedSmugglerShip  = NULL;
69      TradeItem *pobjSelectedTradeItem    = new TradeItem( "Junk", 0, false );
70      TradeItem *pobjSelectedTradeItem02  = new TradeItem( "Junk", 0, false );
71      cout << endl << endl << endl << endl;
72      cout << "Welcome to Smuggler!  Time to create a ship!" << endl << endl;
73
74      boolContinue    = true;
75      while( boolContinue )
76      {
77          cout << "Would you like to create a [U]nique ship or the [D]efault ship?  ";
78          charUserInput = objProcessUserInput.readChar( "UuDd" );
79
80          string strCaptainName                   = "";
81          string strShipName                      = "";
82          unsigned int uintLegalCargoCapacity     = 0;
83          unsigned int uintIllegalCargoCapacity   = 0;
84          unsigned int uintAstros                 = 0;
85
86          switch( charUserInput )
87          {
88              case 'u':
89              case 'U':
90                  {
91                      cout << "What is the captain's name?  ";
92                      strCaptainName  = objProcessUserInput.readString( false, 30 );
93
94                      cout << "What would you like to name the ship?  ";
95                      strShipName     = objProcessUserInput.readString( false, 30 );
96
97                      cout << "How many cargo bays will the ship have for legal trade items?  ";
98                      uintLegalCargoCapacity      = objProcessUserInput.readInt( true, 1, 10 );
99
100                     cout << "How many secret cargo bays will the ship have for illegal trade items?  ";
```

```
                    uintIllegalCargoCapacity    = objProcessUserInput.readInt( true, 1, 10 );

                    cout << "How many astros (currency) will the ship have?  ";
                    uintAstros      = objProcessUserInput.readInt( true, 0, 999999 );

                    pobjSelectedSmugglerShip    = new SmugglerShip( strCaptainName, strShipName, uintLegalCargoCapacity,
                    uintIllegalCargoCapacity, uintAstros );

                    boolContinue    = false;
                    break;
                }

            case 'd':
            case 'D':
                {
                    pobjSelectedSmugglerShip    = new SmugglerShip;

                    boolContinue    = false;
                    break;
                }
        }
    }

    cout << endl;
    cout << "Congratulations, you have just created a ship!" << endl;


    boolContinue    = true;
    while( boolContinue )
    {
        cout << "What would you like to do?" << endl << endl;

        cout << "\tWork with [A]stros" << endl;
        cout << "\tWork with [I]tems" << endl;
        cout << "\tWork with [S]hip" << endl;
        cout << "\t[Q]uit" << endl;

        charUserInput   = objProcessUserInput.readChar( "aAiIsSqQ" );

        switch( charUserInput )
        {
            case 'a':
            case 'A':
                workWithAstros( objProcessUserInput, *pobjSelectedSmugglerShip );
                break;

            case 'i':
            case 'I':
                workWithItems( objProcessUserInput, *pobjSelectedSmugglerShip, *pobjSelectedTradeItem );
                break;
```

```
150
151             case 's':
152             case 'S':
153                 workWithShip( objProcessUserInput, *pobjSelectedSmugglerShip );
154                 break;
155
156             case 'q':
157             case 'Q':
158                 cout << "See ya!" << endl;
159                 boolContinue    = false;
160                 break;
161         }
162     }
163
164     delete pobjSelectedSmugglerShip;
165     delete pobjSelectedTradeItem;
166     delete pobjSelectedTradeItem02;
167
168     return 0;
169 }
170
171 void workWithAstros( CinReader &objProcessUserInput, SmugglerShip &objSelectedSmugglerShip )
172 {
173     char    charUserInput   = '0';
174     int     intUserInput    = 0;
175     cout << "[A]dd astros\t\t[S]pend astros" << endl;
176     cout << "[G]et astros\t\tS[e]t astros" << endl;
177
178     charUserInput       = objProcessUserInput.readChar( "aAsSgGeE" );
179
180     switch( charUserInput )
181     {
182         case 'a':
183         case 'A':
184             cout << "How many astros would you like to add to your ship?  ";
185             intUserInput    = objProcessUserInput.readInt( true, 0, 999999 );
186             objSelectedSmugglerShip.addAstros( intUserInput );
187             cout << intUserInput << " astros have been added to your ship." << endl;
188             break;
189
190         case 's':
191         case 'S':
192             cout << "How many astros would you like to spend?  ";
193             intUserInput    = objProcessUserInput.readInt( true, 0, 999999 );
194
195             if( objSelectedSmugglerShip.spendAstros( intUserInput ) )
196             {
197                 objSelectedSmugglerShip.spendAstros( intUserInput );
198                 cout << "Congrats!  You spent " << intUserInput << " astros!" << endl;
199             }
```

```
200             else
201             {
202                 cout << "You don't have " << intUserInput << " astros!" << endl;
203             }
204             break;
205
206         case 'g':
207         case 'G':
208             cout << "Your ship currently has " << objSelectedSmugglerShip.getAstros() << " astros!" << endl;
209             break;
210
211         case 'e':
212         case 'E':
213             cout << "How many astros would you like to have in your ship?  ";
214             intUserInput    = objProcessUserInput.readInt( true, 0, 999999 );
215             objSelectedSmugglerShip.setAstros( intUserInput );
216             cout << "Your ship now has " << intUserInput << " astros!" << endl;
217             break;
218     }
219 }
220
221 void workWithItems( CinReader &objProcessUserInput, SmugglerShip &objSelectedSmugglerShip, TradeItem &objSelectedTradeItem )
222 {
223     char    charUserInput   = '0';
224     int     intUserInput    = 0;
225     string  strUserInput    = "";
226     bool    boolUserInput   = false;
227     cout << "[A]dd cargo\t\t[C]heck/Select cargo" << endl;
228     cout << "[R]emove cargo\t\t[G]et cargo capacity" << endl;
229     cout << "Com[p]are items" << endl;
230     cout << "Get [N]ame of currently selected cargo" << endl;
231     cout << "[S]et name of currently selected cargo" << endl;
232     cout << "Get item [V]alue" << endl;
233     cout << "Set item Va[l]ue" << endl;
234     cout << "Get Contra[b]and status of selected cargo" << endl;
235     cout << "Set C[o]ntraband status of selected cargo" << endl;
236
237     charUserInput       = objProcessUserInput.readChar( "aAcCrRgGpPnNsSvVlLbBoO" );
238
239     switch( charUserInput )
240     {
241         case 'a':
242         case 'A':
243             cout << "What is the name of the cargo?  ";
244             strUserInput    = objProcessUserInput.readString( false, 30 );
245             cout << "What is the item's value (in astros)?  ";
246             intUserInput    = objProcessUserInput.readInt( true, 0, 999999 );
247             cout << "Is the item considered to be contraband? (Y/N)   ";
248             charUserInput   = objProcessUserInput.readChar( "yYnN" );
249
```

```
            if( charUserInput == 'y' || charUserInput == 'Y' )
            {
                boolUserInput   = true;
                charUserInput   = 'i';
            }
            else
            {
                boolUserInput   = false;
                charUserInput   = 'l';
            }

            if( objSelectedSmugglerShip.addCargo( TradeItem( strUserInput, intUserInput, boolUserInput ),  charUserInput ) )
                cout << "You have added this item to your ship!" << endl;
            else
                cout << "You don't have any more room!" << endl;

            break;

        case 'c':
        case 'C':
            cout << "Would you like to check [l]egal or [i]llegal cargo?  ";
            charUserInput   = objProcessUserInput.readChar( "lLiI" );
            cout << "Which cargo hold would you like to check (1 through " << objSelectedSmugglerShip.getCapacity( charUserInput ) << ")?  ";
            intUserInput    = objProcessUserInput.readInt( true, 1, objSelectedSmugglerShip.getCapacity( charUserInput ) );

            objSelectedTradeItem = objSelectedSmugglerShip.checkCargo( (intUserInput - 1), charUserInput );

            if( objSelectedTradeItem.getItemName() == "Junk" )
                cout << "You only have a bunch of scrap there!" << endl;
            else
                cout << "You have checked the item " << objSelectedTradeItem.getItemName() << "!" << endl;

            break;

        case 'r':
        case 'R':
            cout << "Would you like to remove [l]egal or [i]llegal cargo?  ";
            charUserInput   = objProcessUserInput.readChar( "lLiI" );
            cout << "Which cargo hold would you like to remove (1 through " << objSelectedSmugglerShip.getCapacity( charUserInput ) << ")?  ";
            intUserInput    = objProcessUserInput.readInt( true, 1, objSelectedSmugglerShip.getCapacity( charUserInput ) );

            objSelectedTradeItem = objSelectedSmugglerShip.checkCargo( (intUserInput - 1), charUserInput );

            if( objSelectedTradeItem.getItemName() == "Junk" )
                cout << "You only have a bunch of scrap there!" << endl;
            else
                cout << "You have removed the item " << objSelectedTradeItem.getItemName() << "!" << endl;

```

```
            break;

        case 'g':
        case 'G':
            cout << "Would you like to get the capacity for [l]egal or [i]llegal cargo?  ";
            charUserInput   = objProcessUserInput.readChar( "lLiI" );

            cout << "Your ship can hold " << objSelectedSmugglerShip.getCapacity( charUserInput ) << " of that kind." << endl;
            break;

        case 'n':
        case 'N':
            cout << "The name of the currently selected item is " << objSelectedTradeItem.getItemName() << "!" << endl;

            break;

        case 'p':
        case 'P':
            cout << "Would you like to compare this item with [l]egal or [i]llegal cargo?  ";
            charUserInput   = objProcessUserInput.readChar( "lLiI" );
            cout << "Which cargo hold contains the item for comparison (1 through " << objSelectedSmugglerShip.getCapacity(
            charUserInput ) << ")?  ";
            intUserInput    = objProcessUserInput.readInt( true, 1, objSelectedSmugglerShip.getCapacity( charUserInput ) );

            if( objSelectedTradeItem == objSelectedSmugglerShip.checkCargo( (intUserInput - 1), charUserInput ) )
                cout << "The items are the same!" << endl;
            else
                cout << "The items are not the same!" << endl;

            break;

        case 's':
        case 'S':
            cout << "What is the new name for the currently selected item?  "  ;
            strUserInput    = objProcessUserInput.readString( false, 30 );
            objSelectedTradeItem.setItemName( strUserInput );

            cout << "Your item has been renamed to """ << strUserInput << """!" << endl;

            break;

        case 'v':
        case 'V':
            cout << "The value of the currently selected item is " << objSelectedTradeItem.getItemValue() << " astros!" << endl;

            break;

        case 'l':
        case 'L':
            cout << "What is the new value for the currently selected item?  "  ;
```

```
347             intUserInput    = objProcessUserInput.readInt( true, 0, 9999999 );
348             objSelectedTradeItem.setItemValue( intUserInput );
349
350             cout << "Your item has a new value of " << intUserInput << " astros!" << endl;
351             break;
352
353         case 'b':
354         case 'B':
355             if( objSelectedTradeItem.getIsContraband() )
356                 cout << "The selected item is contraband!" << endl;
357             else
358                 cout << "The selected item is not contraband!" << endl;
359             break;
360
361         case 'o':
362         case 'O':
363             cout << "Should the selected item be made to be [l]egal or [i]llegal cargo?  ";
364             charUserInput   = objProcessUserInput.readChar( "lLiI" );
365
366             if( charUserInput == 'l' || charUserInput == 'L' )
367                 objSelectedTradeItem.setIsContraband( false );
368             else
369                 objSelectedTradeItem.setIsContraband( true );
370
371             break;
372     }
373 }
374
375 void workWithShip( CinReader &objProcessUserInput, SmugglerShip &objSelectedSmugglerShip )
376 {
377     char    charUserInput   = '0';
378     int     intUserInput    = 0;
379     string  strUserInput    = "";
380     bool    boolUserInput   = false;
381     cout << "Get [C]aptain Name\t\tSet Ca[p]tain Name" << endl;
382     cout << "Get [S]hip Name\t\t\tSet Sh[i]p name" << endl;
383
384     charUserInput       = objProcessUserInput.readChar( "cCpPsSiI" );
385
386     switch( charUserInput )
387     {
388         case 'c':
389         case 'C':
390             cout << "The captain of this ship is named " << objSelectedSmugglerShip.getCaptainName() << "!" << endl;
391             break;
392
393         case 'p':
394         case 'P':
395             cout << "What will be the captain's new name?  "  ;
396             strUserInput    = objProcessUserInput.readString( false, 30 );
```

```
            objSelectedSmugglerShip.setCaptainName( strUserInput );

            cout << "The captain's name has been changed to """ << strUserInput << """!" << endl;
            cout << "He loves it and asks if there is anything else he can do for you today!" << endl;

            break;

        case 's':
        case 'S':
            cout << "The ship's name is " << objSelectedSmugglerShip.getShipName() << "!" << endl;
            break;

        case 'i':
        case 'I':
            cout << "What will be the ship's new name?  "   ;
            strUserInput    = objProcessUserInput.readString( false, 30 );
            objSelectedSmugglerShip.setShipName( strUserInput );

            cout << "The ship's name has been changed to """ << strUserInput << """!" << endl;
            break;
    }
}

/*
 * Unit testing functions. Do not alter.
 */

void unittest ()
{
    cout << "\nSTARTING UNIT TEST\n\n";

    SmugglerShip s("Captain Peabody", "SS Unit Test", 3, 3, 100);

    cout << "** TESTING SMUGGLERSHIP **\n\n";

    try {
        btassert<bool>(s.getCaptainName() == "Captain Peabody");
        cout << "Passed TEST 1: SmugglerShip getCaptainName()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 1 SmugglerShip getCaptainName() #\n";
    }

    try {
        btassert<bool>(s.getShipName() == "SS Unit Test");
        cout << "Passed TEST 2: SmugglerShip getShipName()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 2 SmugglerShip getShipName() #\n";
    }

    s.setCaptainName("CAPTAIN PEABODY");
```

```
    try {
        btassert<bool>(s.getCaptainName() == "CAPTAIN PEABODY");
        cout << "Passed TEST 3: SmugglerShip setCaptainName()/getCaptainName()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 3 SmugglerShip setCaptainName()/getCaptainName() #\n";
    }

    s.setShipName("SS UNIT TEST");
    try {
        btassert<bool>(s.getShipName() == "SS UNIT TEST");
        cout << "Passed TEST 4: SmugglerShip setShipName()/getShipName()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 4 SmugglerShip setShipName()/getShipName() #\n";
    }

    try {
        btassert<bool>(s.getAstros() == 100);
        cout << "Passed TEST 5: SmugglerShip getAstros()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 5 SmugglerShip getAstros() #\n";
    }

    s.addAstros(15);
    try {
        btassert<bool>(s.getAstros() == 115);
        cout << "Passed TEST 6: SmugglerShip addAstros(15)/getAstros()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 6 SmugglerShip addAstros(15)/getAstros() #\n";
    }

    try {
        btassert<bool>(s.spendAstros(116) == false);
        cout << "Passed TEST 7: SmugglerShip spendAstros(116)\n";
    } catch (bool b) {
        cout << "# FAILED TEST 7 SmugglerShip spendAstros(116) #\n";
    }

    s.spendAstros(114);
    try {
        btassert<bool>(s.getAstros() == 1);
        cout << "Passed TEST 8: SmugglerShip spendAstros(114)/getAstros()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 8 SmugglerShip spendAstros(114)/getAstros() #\n";
    }

    try {
        btassert<bool>(s.checkCargo(0, 'L') == TradeItem());
        cout << "Passed TEST 9: SmugglerShip checkCargo(0, 'L')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 9 SmugglerShip checkCargo(0, 'L') #\n";
```

```
    }

    try {
        btassert<bool>(s.removeCargo(0, 'L') == TradeItem());
        cout << "Passed TEST 10: SmugglerShip removeCargo(0, 'L')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 10 SmugglerShip removeCargo(0, 'L') #\n";
    }

    try {
        btassert<bool>(s.checkCargo(0, 'I') == TradeItem());
        cout << "Passed TEST 11: SmugglerShip checkCargo(0, 'I')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 11 SmugglerShip checkCargo(0, 'I') #\n";
    }

    try {
        btassert<bool>(s.removeCargo(0, 'L') == TradeItem());
        cout << "Passed TEST 12: SmugglerShip removeCargo(0, 'I')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 12 SmugglerShip removeCargo(0, 'I') #\n";
    }

    s.addCargo(TradeItem("RED", 1), 'L');
    s.addCargo(TradeItem("GREEN", 2), 'L');
    s.addCargo(TradeItem("BLUE", 3), 'L');
    try {
        btassert<bool>(s.addCargo(TradeItem("GOLD", 4), 'L') == false);
        cout << "Passed TEST 13: SmugglerShip addCargo(\"GOLD\",'L')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 13 SmugglerShip addCargo(\"GOLD\",'L') #\n";
    }

    try {
        btassert<bool>(s.checkCargo(1, 'L') == TradeItem("GREEN", 2));
        cout << "Passed TEST 14: SmugglerShip checkCargo(1, 'L')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 14 SmugglerShip checkCargo(1, 'L') #\n";
    }

    try {
        btassert<bool>(s.removeCargo(2, 'L') == TradeItem("BLUE", 3));
        cout << "Passed TEST 15: SmugglerShip removeCargo(2, 'L')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 15 SmugglerShip removeCargo(2, 'L') #\n";
    }

    try {
        btassert<bool>(s.removeCargo(2, 'L') == TradeItem());
        cout << "Passed TEST 16: SmugglerShip removeCargo(2, 'L')\n";
```

```
    } catch (bool b) {
        cout << "# FAILED TEST 16 SmugglerShip removeCargo(2, 'L') #\n";
    }

    try {
        btassert<bool>(s.checkCargo(2, 'L') == TradeItem());
        cout << "Passed TEST 17: SmugglerShip checkCargo(2, 'L')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 17 SmugglerShip checkCargo(2, 'L') #\n";
    }

    try {
        btassert<bool>(s.addCargo(TradeItem("PURPLE", 5), 'L') == true);
        cout << "Passed TEST 18: SmugglerShip addCargo(\"PURPLE\", 'L')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 18 SmugglerShip addCargo(\"PURPLE\", 'L') #\n";
    }

    s.addCargo(TradeItem("RED", 1), 'I');
    s.addCargo(TradeItem("GREEN", 2), 'I');
    s.addCargo(TradeItem("BLUE", 3), 'I');
    try {
        btassert<bool>(s.addCargo(TradeItem("GOLD", 4), 'I') == false);
        cout << "Passed TEST 19: SmugglerShip addCargo(\"GOLD\",'I')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 19 SmugglerShip addCargo(\"GOLD\",'I') #\n";
    }

    try {
        btassert<bool>(s.checkCargo(1, 'I') == TradeItem("GREEN", 2));
        cout << "Passed TEST 20: SmugglerShip checkCargo(1, 'I')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 20 SmugglerShip checkCargo(1, 'I') #\n";
    }

    try {
        btassert<bool>(s.removeCargo(2, 'I') == TradeItem("BLUE", 3));
        cout << "Passed TEST 21: SmugglerShip removeCargo(2, 'I')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 21 SmugglerShip removeCargo(2, 'I') #\n";
    }

    try {
        btassert<bool>(s.removeCargo(2, 'I') == TradeItem());
        cout << "Passed TEST 22: SmugglerShip removeCargo(2, 'I')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 22 SmugglerShip removeCargo(2, 'I') #\n";
    }

    try {
```

```
        btassert<bool>(s.checkCargo(2, 'I') == TradeItem());
        cout << "Passed TEST 23: SmugglerShip checkCargo(2, 'I')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 23 SmugglerShip checkCargo(2, 'I') #\n";
    }

    try {
        btassert<bool>(s.addCargo(TradeItem("PURPLE", 5), 'I') == true);
        cout << "Passed TEST 24: SmugglerShip addCargo(\"PURPLE\", 'I')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 24 SmugglerShip addCargo(\"PURPLE\", 'I') #\n";
    }

    try {
        btassert<bool>(s.getCapacity('L') == 3);
        cout << "Passed TEST 25: SmugglerShip getCapacity('L')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 25 SmugglerShip getCapacity('L') #\n";
    }

    try {
        btassert<bool>(s.getCapacity('I') == 3);
        cout << "Passed TEST 26: SmugglerShip getCapacity('I')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 26 SmugglerShip getCapacity('I') #\n";
    }

    SmugglerShip s1;

    try {
        btassert<bool>(s1.getCaptainName() == "No Name");
        cout << "Passed TEST 27: SmugglerShip getCaptainName()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 27 SmugglerShip getCaptainName() #\n";
    }

    try {
        btassert<bool>(s1.getShipName() == "SS Smuggler");
        cout << "Passed TEST 28: SmugglerShip getShipName()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 28 SmugglerShip getShipName() #\n";
    }

    try {
        btassert<bool>(s1.getCapacity('L') == 5);
        cout << "Passed TEST 29: SmugglerShip getCapacity('L')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 29 SmugglerShip getCapacity('L') #\n";
    }

```

```
    try {
        btassert<bool>(s1.getCapacity('I') == 3);
        cout << "Passed TEST 30: SmugglerShip getCapacity('I')\n";
    } catch (bool b) {
        cout << "# FAILED TEST 30 SmugglerShip getCapacity('I') #\n";
    }

    try {
        btassert<bool>(s1.getAstros() == 100);
        cout << "Passed TEST 31: SmugglerShip getAstros()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 31 SmugglerShip getAstros() #\n";
    }

    cout << "\n** TESTING TRADE ITEM **\n\n";

    TradeItem i;

    try {
        btassert<bool>(i.getItemName() == "EMPTY");
        cout << "Passed TEST 32: TradeItem getItemName()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 32 TradeItem getItemName() #\n";
    }

    try {
        btassert<bool>(i.getItemValue() == 0);
        cout << "Passed TEST 33: TradeItem getItemValue()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 33 TradeItem getItemValue() #\n";
    }

    try {
        btassert<bool>(i.getIsContraband() == false);
        cout << "Passed TEST 34: TradeItem getIsContraband()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 34 TradeItem getIsContraband() #\n";
    }

    i.setItemName("GREEN");
    try {
        btassert<bool>(i.getItemName() == "GREEN");
        cout << "Passed TEST 35: TradeItem setItemName()/getItemName()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 35 TradeItem setItemName()/getItemName() #\n";
    }

    i.setItemValue(99);
    try {
        btassert<bool>(i.getItemValue() == 99);
```

```
        cout << "Passed TEST 36: TradeItem setItemValue(99)/getItemValue()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 36 TradeItem setItemValue(99)/getItemValue() #\n";
    }

    i.setIsContraband(true);
    try {
        btassert<bool>(i.getIsContraband() == true);
        cout << "Passed TEST 37: TradeItem setIsContraband(true)/getIsContraband()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 37 TradeItem setIsContraband(true)/getIsContraband() #\n";
    }

    TradeItem i1("RED", 1, true);

    try {
        btassert<bool>(i1.getItemName() == "RED");
        cout << "Passed TEST 38: TradeItem getItemName()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 38 TradeItem getItemName() #\n";
    }

    try {
        btassert<bool>(i1.getItemValue() == 1);
        cout << "Passed TEST 39: TradeItem getItemValue()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 39 TradeItem getItemValue() #\n";
    }

    try {
        btassert<bool>(i1.getIsContraband() == true);
        cout << "Passed TEST 40: TradeItem getIsContraband()\n";
    } catch (bool b) {
        cout << "# FAILED TEST 40 TradeItem getIsContraband() #\n";
    }

    cout << "\nUNIT TEST COMPLETE\n\n";
}

template <typename X, typename A>
void btassert (A assertion)
{
    if (!assertion)
        throw X();
}
```