

```

1  /*
2  * Program name: Smuggler (Programming Project 02)
3  * Program description: Driver for the smuggler game which handles the creation and management
4  *   of SmugglerShip and TradeItem instances. Provides a simple menu so that the user can
5  *   interact with the functions within these objects.
6  *
7  * Programmer: Chad Philip Johnson
8  * Date created: February 21st, 2013
9  * Last date modified: May 10th, 2013
10 *
11 * Sources Used:
12 *   CinReader.h
13 *       - for handling user input and to ensure that valid values are passed into the program
14 *   smugglership.h
15 *       - for accomodating created instances of the SmugglerShip class
16 *   tradeitem.h
17 *       - for accomodating created instances of the TradeItem class
18 */
19
20 #pragma once
21
22 #include <cstdlib>
23 #include <iostream>
24 #include <string>
25
26 #include "smugglership.h"
27 #include "tradeitem.h"
28 #include "CinReader.h"
29
30 using namespace std;
31
32 /* for unit testing -- do not alter */
33 template <typename X, typename A>
34 void btassert(A assertion);
35 void unittest ();
36
37 /**
38 * Menu system for interactive test. Allows the user to work with the astros aboard a SmugglerShip object.
39 * @param objProcessUserInput Instance of CinReader to handle user input.
40 * @param objSelectedSmugglerShip The instance of SmugglerShip for the user to interact with.
41 */
42 void workWithAstros( CinReader &objProcessUserInput, SmugglerShip &objSelectedSmugglerShip);
43
44 /**
45 * Menu system for interactive test. Allows the user to work with the trade items aboard a SmugglerShip object and the ship's
46 * cargo capacities.
47 * @param objProcessUserInput Instance of CinReader to handle user input.
48 * @param objSelectedSmugglerShip The instance of SmugglerShip for the user to interact with.
49 * @param objSelectedTradeItem The instance of TradeItem for the user to interact with.
50 */

```

```

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? ";

```

```

101         uintIllegalCargoCapacity    = objProcessUserInput.readInt( true, 1, 10 );
102
103         cout << "How many astros (currency) will the ship have? ";
104         uintAstros                    = objProcessUserInput.readInt( true, 0, 999999 );
105
106         pObjSelectedSmugglerShip      = new SmugglerShip( strCaptainName, strShipName, uintLegalCargoCapacity,
107                                                         uintIllegalCargoCapacity, uintAstros );
108
109         boolContinue                  = false;
110         break;
111     }
112     case 'd':
113     case 'D':
114     {
115         pObjSelectedSmugglerShip      = new SmugglerShip;
116
117         boolContinue                  = false;
118         break;
119     }
120 }
121 }
122
123 cout << endl;
124 cout << "Congratulations, you have just created a ship!" << endl;
125
126
127 boolContinue    = true;
128 while( boolContinue )
129 {
130     cout << "What would you like to do?" << endl << endl;
131
132     cout << "\tWork with [A]stros" << endl;
133     cout << "\tWork with [I]tems" << endl;
134     cout << "\tWork with [S]hip" << endl;
135     cout << "\t[Q]uit" << endl;
136
137     charUserInput    = objProcessUserInput.readChar( "aAiIsSqQ" );
138
139     switch( charUserInput )
140     {
141         case 'a':
142         case 'A':
143             workWithAstros( objProcessUserInput, *pObjSelectedSmugglerShip );
144             break;
145
146         case 'i':
147         case 'I':
148             workWithItems( objProcessUserInput, *pObjSelectedSmugglerShip, *pObjSelectedTradeItem );
149             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\t[S][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

```

```

250     if( charUserInput == 'y' || charUserInput == 'Y' )
251     {
252         boolUserInput    = true;
253         charUserInput    = 'i';
254     }
255     else
256     {
257         boolUserInput    = false;
258         charUserInput    = 'l';
259     }
260
261     if( objSelectedSmugglerShip.addCargo( TradeItem( strUserInput, intUserInput, boolUserInput ), charUserInput ) )
262         cout << "You have added this item to your ship!" << endl;
263     else
264         cout << "You don't have any more room!" << endl;
265
266     break;
267
268 case 'c':
269 case 'C':
270     cout << "Would you like to check [l]egal or [i]llegal cargo? ";
271     charUserInput = objProcessUserInput.readChar( "lLiI" );
272     cout << "Which cargo hold would you like to check (1 through " << objSelectedSmugglerShip.getCapacity( charUserInput
273 ) << ")? ";
274     intUserInput = objProcessUserInput.readInt( true, 1, objSelectedSmugglerShip.getCapacity( charUserInput ) );
275
276     objSelectedTradeItem = objSelectedSmugglerShip.checkCargo( (intUserInput - 1), charUserInput );
277
278     if( objSelectedTradeItem.getItemName() == "Junk" )
279         cout << "You only have a bunch of scrap there!" << endl;
280     else
281         cout << "You have checked the item " << objSelectedTradeItem.getItemName() << "!" << endl;
282
283     break;
284
285 case 'r':
286 case 'R':
287     cout << "Would you like to remove [l]egal or [i]llegal cargo? ";
288     charUserInput = objProcessUserInput.readChar( "lLiI" );
289     cout << "Which cargo hold would you like to remove (1 through " << objSelectedSmugglerShip.getCapacity( charUserInput
290 ) << ")? ";
291     intUserInput = objProcessUserInput.readInt( true, 1, objSelectedSmugglerShip.getCapacity( charUserInput ) );
292
293     objSelectedTradeItem = objSelectedSmugglerShip.checkCargo( (intUserInput - 1), charUserInput );
294
295     if( objSelectedTradeItem.getItemName() == "Junk" )
296         cout << "You only have a bunch of scrap there!" << endl;
297     else
298         cout << "You have removed the item " << objSelectedTradeItem.getItemName() << "!" << endl;

```

```

298         break;
299
300     case 'g':
301     case 'G':
302         cout << "Would you like to get the capacity for [l]legal or [i]llegal cargo? ";
303         charUserInput = objProcessUserInput.readChar( "lLiI" );
304
305         cout << "Your ship can hold " << objSelectedSmugglerShip.getCapacity( charUserInput ) << " of that kind." << endl;
306         break;
307
308     case 'n':
309     case 'N':
310         cout << "The name of the currently selected item is " << objSelectedTradeItem.getItemName() << "!" << endl;
311
312         break;
313
314     case 'p':
315     case 'P':
316         cout << "Would you like to compare this item with [l]legal or [i]llegal cargo? ";
317         charUserInput = objProcessUserInput.readChar( "lLiI" );
318         cout << "Which cargo hold contains the item for comparison (1 through " << objSelectedSmugglerShip.getCapacity(
319             charUserInput ) << ")? ";
320         intUserInput = objProcessUserInput.readInt( true, 1, objSelectedSmugglerShip.getCapacity( charUserInput ) );
321
322         if( objSelectedTradeItem == objSelectedSmugglerShip.checkCargo( (intUserInput - 1), charUserInput ) )
323             cout << "The items are the same!" << endl;
324         else
325             cout << "The items are not the same!" << endl;
326
327         break;
328
329     case 's':
330     case 'S':
331         cout << "What is the new name for the currently selected item? ";
332         strUserInput = objProcessUserInput.readString( false, 30 );
333         objSelectedTradeItem.setItemName( strUserInput );
334
335         cout << "Your item has been renamed to "" << strUserInput << ""!" << endl;
336
337         break;
338
339     case 'v':
340     case 'V':
341         cout << "The value of the currently selected item is " << objSelectedTradeItem.getItemValue() << " astros!" << endl;
342
343         break;
344
345     case 'l':
346     case 'L':
347         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]legal 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 );

```



```

397         objSelectedSmugglerShip.setCaptainName( strUserInput );
398
399         cout << "The captain's name has been changed to "" << strUserInput << ""!" << endl;
400         cout << "He loves it and asks if there is anything else he can do for you today!" << endl;
401
402         break;
403
404     case 's':
405     case 'S':
406         cout << "The ship's name is " << objSelectedSmugglerShip.getShipName() << "!" << endl;
407         break;
408
409     case 'i':
410     case 'I':
411         cout << "What will be the ship's new name? " ;
412         strUserInput = objProcessUserInput.readString( false, 30 );
413         objSelectedSmugglerShip.setShipName( strUserInput );
414
415         cout << "The ship's name has been changed to "" << strUserInput << ""!" << endl;
416         break;
417     }
418 }
419
420 /*
421  * Unit testing functions. Do not alter.
422  */
423
424 void unittest ()
425 {
426     cout << "\nSTARTING UNIT TEST\n\n";
427
428     SmugglerShip s("Captain Peabody", "SS Unit Test", 3, 3, 100);
429
430     cout << "*** TESTING SMUGGLERSHIP **\n\n";
431
432     try {
433         btassert<bool>(s.getCaptainName() == "Captain Peabody");
434         cout << "Passed TEST 1: SmugglerShip getCaptainName()\n";
435     } catch (bool b) {
436         cout << "# FAILED TEST 1 SmugglerShip getCaptainName() #\n";
437     }
438
439     try {
440         btassert<bool>(s.getShipName() == "SS Unit Test");
441         cout << "Passed TEST 2: SmugglerShip getShipName()\n";
442     } catch (bool b) {
443         cout << "# FAILED TEST 2 SmugglerShip getShipName() #\n";
444     }
445
446     s.setCaptainName("CAPTAIN PEABODY");

```

```

447     try {
448         btassert<bool>(s.getCaptainName() == "CAPTAIN PEABODY");
449         cout << "Passed TEST 3: SmugglerShip setCaptainName()/getCaptainName()\n";
450     } catch (bool b) {
451         cout << "# FAILED TEST 3 SmugglerShip setCaptainName()/getCaptainName() #\n";
452     }
453
454     s.setShipName("SS UNIT TEST");
455     try {
456         btassert<bool>(s.getShipName() == "SS UNIT TEST");
457         cout << "Passed TEST 4: SmugglerShip setShipName()/getShipName()\n";
458     } catch (bool b) {
459         cout << "# FAILED TEST 4 SmugglerShip setShipName()/getShipName() #\n";
460     }
461
462     try {
463         btassert<bool>(s.getAstros() == 100);
464         cout << "Passed TEST 5: SmugglerShip getAstros()\n";
465     } catch (bool b) {
466         cout << "# FAILED TEST 5 SmugglerShip getAstros() #\n";
467     }
468
469     s.addAstros(15);
470     try {
471         btassert<bool>(s.getAstros() == 115);
472         cout << "Passed TEST 6: SmugglerShip addAstros(15)/getAstros()\n";
473     } catch (bool b) {
474         cout << "# FAILED TEST 6 SmugglerShip addAstros(15)/getAstros() #\n";
475     }
476
477     try {
478         btassert<bool>(s.spendAstros(116) == false);
479         cout << "Passed TEST 7: SmugglerShip spendAstros(116)\n";
480     } catch (bool b) {
481         cout << "# FAILED TEST 7 SmugglerShip spendAstros(116) #\n";
482     }
483
484     s.spendAstros(114);
485     try {
486         btassert<bool>(s.getAstros() == 1);
487         cout << "Passed TEST 8: SmugglerShip spendAstros(114)/getAstros()\n";
488     } catch (bool b) {
489         cout << "# FAILED TEST 8 SmugglerShip spendAstros(114)/getAstros() #\n";
490     }
491
492     try {
493         btassert<bool>(s.checkCargo(0, 'L') == TradeItem());
494         cout << "Passed TEST 9: SmugglerShip checkCargo(0, 'L')\n";
495     } catch (bool b) {
496         cout << "# FAILED TEST 9 SmugglerShip checkCargo(0, 'L') #\n";

```

```

497     }
498
499     try {
500         btassert<bool>(s.removeCargo(0, 'L') == TradeItem());
501         cout << "Passed TEST 10: SmugglerShip removeCargo(0, 'L')\n";
502     } catch (bool b) {
503         cout << "# FAILED TEST 10 SmugglerShip removeCargo(0, 'L') #\n";
504     }
505
506     try {
507         btassert<bool>(s.checkCargo(0, 'I') == TradeItem());
508         cout << "Passed TEST 11: SmugglerShip checkCargo(0, 'I')\n";
509     } catch (bool b) {
510         cout << "# FAILED TEST 11 SmugglerShip checkCargo(0, 'I') #\n";
511     }
512
513     try {
514         btassert<bool>(s.removeCargo(0, 'L') == TradeItem());
515         cout << "Passed TEST 12: SmugglerShip removeCargo(0, 'I')\n";
516     } catch (bool b) {
517         cout << "# FAILED TEST 12 SmugglerShip removeCargo(0, 'I') #\n";
518     }
519
520     s.addCargo(TradeItem("RED", 1), 'L');
521     s.addCargo(TradeItem("GREEN", 2), 'L');
522     s.addCargo(TradeItem("BLUE", 3), 'L');
523     try {
524         btassert<bool>(s.addCargo(TradeItem("GOLD", 4), 'L') == false);
525         cout << "Passed TEST 13: SmugglerShip addCargo(\"GOLD\", 'L')\n";
526     } catch (bool b) {
527         cout << "# FAILED TEST 13 SmugglerShip addCargo(\"GOLD\", 'L') #\n";
528     }
529
530     try {
531         btassert<bool>(s.checkCargo(1, 'L') == TradeItem("GREEN", 2));
532         cout << "Passed TEST 14: SmugglerShip checkCargo(1, 'L')\n";
533     } catch (bool b) {
534         cout << "# FAILED TEST 14 SmugglerShip checkCargo(1, 'L') #\n";
535     }
536
537     try {
538         btassert<bool>(s.removeCargo(2, 'L') == TradeItem("BLUE", 3));
539         cout << "Passed TEST 15: SmugglerShip removeCargo(2, 'L')\n";
540     } catch (bool b) {
541         cout << "# FAILED TEST 15 SmugglerShip removeCargo(2, 'L') #\n";
542     }
543
544     try {
545         btassert<bool>(s.removeCargo(2, 'L') == TradeItem());
546         cout << "Passed TEST 16: SmugglerShip removeCargo(2, 'L')\n";

```

```

547     } catch (bool b) {
548         cout << "# FAILED TEST 16 SmugglerShip removeCargo(2, 'L') #\n";
549     }
550
551     try {
552         btassert<bool>(s.checkCargo(2, 'L') == TradeItem());
553         cout << "Passed TEST 17: SmugglerShip checkCargo(2, 'L')\n";
554     } catch (bool b) {
555         cout << "# FAILED TEST 17 SmugglerShip checkCargo(2, 'L') #\n";
556     }
557
558     try {
559         btassert<bool>(s.addCargo(TradeItem("PURPLE", 5), 'L') == true);
560         cout << "Passed TEST 18: SmugglerShip addCargo(\"PURPLE\", 'L')\n";
561     } catch (bool b) {
562         cout << "# FAILED TEST 18 SmugglerShip addCargo(\"PURPLE\", 'L') #\n";
563     }
564
565     s.addCargo(TradeItem("RED", 1), 'I');
566     s.addCargo(TradeItem("GREEN", 2), 'I');
567     s.addCargo(TradeItem("BLUE", 3), 'I');
568     try {
569         btassert<bool>(s.addCargo(TradeItem("GOLD", 4), 'I') == false);
570         cout << "Passed TEST 19: SmugglerShip addCargo(\"GOLD\", 'I')\n";
571     } catch (bool b) {
572         cout << "# FAILED TEST 19 SmugglerShip addCargo(\"GOLD\", 'I') #\n";
573     }
574
575     try {
576         btassert<bool>(s.checkCargo(1, 'I') == TradeItem("GREEN", 2));
577         cout << "Passed TEST 20: SmugglerShip checkCargo(1, 'I')\n";
578     } catch (bool b) {
579         cout << "# FAILED TEST 20 SmugglerShip checkCargo(1, 'I') #\n";
580     }
581
582     try {
583         btassert<bool>(s.removeCargo(2, 'I') == TradeItem("BLUE", 3));
584         cout << "Passed TEST 21: SmugglerShip removeCargo(2, 'I')\n";
585     } catch (bool b) {
586         cout << "# FAILED TEST 21 SmugglerShip removeCargo(2, 'I') #\n";
587     }
588
589     try {
590         btassert<bool>(s.removeCargo(2, 'I') == TradeItem());
591         cout << "Passed TEST 22: SmugglerShip removeCargo(2, 'I')\n";
592     } catch (bool b) {
593         cout << "# FAILED TEST 22 SmugglerShip removeCargo(2, 'I') #\n";
594     }
595
596     try {

```

```

597         btassert<bool>(s.checkCargo(2, 'I') == TradeItem());
598         cout << "Passed TEST 23: SmugglerShip checkCargo(2, 'I')\n";
599     } catch (bool b) {
600         cout << "# FAILED TEST 23 SmugglerShip checkCargo(2, 'I') #\n";
601     }
602
603     try {
604         btassert<bool>(s.addCargo(TradeItem("PURPLE", 5), 'I') == true);
605         cout << "Passed TEST 24: SmugglerShip addCargo(\"PURPLE\", 'I')\n";
606     } catch (bool b) {
607         cout << "# FAILED TEST 24 SmugglerShip addCargo(\"PURPLE\", 'I') #\n";
608     }
609
610     try {
611         btassert<bool>(s.getCapacity('L') == 3);
612         cout << "Passed TEST 25: SmugglerShip getCapacity('L')\n";
613     } catch (bool b) {
614         cout << "# FAILED TEST 25 SmugglerShip getCapacity('L') #\n";
615     }
616
617     try {
618         btassert<bool>(s.getCapacity('I') == 3);
619         cout << "Passed TEST 26: SmugglerShip getCapacity('I')\n";
620     } catch (bool b) {
621         cout << "# FAILED TEST 26 SmugglerShip getCapacity('I') #\n";
622     }
623
624     SmugglerShip s1;
625
626     try {
627         btassert<bool>(s1.getCaptainName() == "No Name");
628         cout << "Passed TEST 27: SmugglerShip getCaptainName()\n";
629     } catch (bool b) {
630         cout << "# FAILED TEST 27 SmugglerShip getCaptainName() #\n";
631     }
632
633     try {
634         btassert<bool>(s1.getShipName() == "SS Smuggler");
635         cout << "Passed TEST 28: SmugglerShip getShipName()\n";
636     } catch (bool b) {
637         cout << "# FAILED TEST 28 SmugglerShip getShipName() #\n";
638     }
639
640     try {
641         btassert<bool>(s1.getCapacity('L') == 5);
642         cout << "Passed TEST 29: SmugglerShip getCapacity('L')\n";
643     } catch (bool b) {
644         cout << "# FAILED TEST 29 SmugglerShip getCapacity('L') #\n";
645     }
646

```

```

647     try {
648         btassert<bool>(s1.getCapacity('I') == 3);
649         cout << "Passed TEST 30: SmugglerShip getCapacity('I')\n";
650     } catch (bool b) {
651         cout << "# FAILED TEST 30 SmugglerShip getCapacity('I') #\n";
652     }
653
654     try {
655         btassert<bool>(s1.getAstros() == 100);
656         cout << "Passed TEST 31: SmugglerShip getAstros()\n";
657     } catch (bool b) {
658         cout << "# FAILED TEST 31 SmugglerShip getAstros() #\n";
659     }
660
661     cout << "\n** TESTING TRADE ITEM **\n\n";
662
663     TradeItem i;
664
665     try {
666         btassert<bool>(i.getItemName() == "EMPTY");
667         cout << "Passed TEST 32: TradeItem getItemName()\n";
668     } catch (bool b) {
669         cout << "# FAILED TEST 32 TradeItem getItemName() #\n";
670     }
671
672     try {
673         btassert<bool>(i.getItemValue() == 0);
674         cout << "Passed TEST 33: TradeItem getItemValue()\n";
675     } catch (bool b) {
676         cout << "# FAILED TEST 33 TradeItem getItemValue() #\n";
677     }
678
679     try {
680         btassert<bool>(i.getIsContraband() == false);
681         cout << "Passed TEST 34: TradeItem getIsContraband()\n";
682     } catch (bool b) {
683         cout << "# FAILED TEST 34 TradeItem getIsContraband() #\n";
684     }
685
686     i.setItemName("GREEN");
687     try {
688         btassert<bool>(i.getItemName() == "GREEN");
689         cout << "Passed TEST 35: TradeItem setItemName()/getItemName()\n";
690     } catch (bool b) {
691         cout << "# FAILED TEST 35 TradeItem setItemName()/getItemName() #\n";
692     }
693
694     i.setItemValue(99);
695     try {
696         btassert<bool>(i.getItemValue() == 99);

```

```

697     cout << "Passed TEST 36: TradeItem setItemValue(99)/getItemValue()\n";
698 } catch (bool b) {
699     cout << "# FAILED TEST 36 TradeItem setItemValue(99)/getItemValue() #\n";
700 }
701
702 i.setIsContraband(true);
703 try {
704     btassert<bool>(i.getIsContraband() == true);
705     cout << "Passed TEST 37: TradeItem setIsContraband(true)/getIsContraband()\n";
706 } catch (bool b) {
707     cout << "# FAILED TEST 37 TradeItem setIsContraband(true)/getIsContraband() #\n";
708 }
709
710 TradeItem i1("RED", 1, true);
711
712 try {
713     btassert<bool>(i1.getItemName() == "RED");
714     cout << "Passed TEST 38: TradeItem getItemName()\n";
715 } catch (bool b) {
716     cout << "# FAILED TEST 38 TradeItem getItemName() #\n";
717 }
718
719 try {
720     btassert<bool>(i1.getItemValue() == 1);
721     cout << "Passed TEST 39: TradeItem getItemValue()\n";
722 } catch (bool b) {
723     cout << "# FAILED TEST 39 TradeItem getItemValue() #\n";
724 }
725
726 try {
727     btassert<bool>(i1.getIsContraband() == true);
728     cout << "Passed TEST 40: TradeItem getIsContraband()\n";
729 } catch (bool b) {
730     cout << "# FAILED TEST 40 TradeItem getIsContraband() #\n";
731 }
732
733 cout << "\nUNIT TEST COMPLETE\n\n";
734 }
735
736 template <typename X, typename A>
737 void btassert (A assertion)
738 {
739     if (!assertion)
740         throw X();
741 }
742
743

```