

```
1 #include "Headers/Destructor.hpp"
2 #include <cmath>
3
4 unsigned int Destroyer::serialNumberCounter = 0;
5
6 unsigned int Destroyer::getNextSerialNumber() {
7     return ++serialNumberCounter;
8 }
9
10 Destroyer::Destroyer(double currentCapacity) : CargoShip(
11     getNextSerialNumber(), currentCapacity) {
12     if(currentCapacity < 0 || currentCapacity > this->getMaxCapacity()
13         ()) {
14         throw std::invalid_argument("Capacity must be greater than 0 and
15             less than or equal to max capacity");
16     }
17 }
18
19
20 double Destroyer::getMaxCapacity() const {
21     return 250 * pow(10, 3);
22 }
23
24 std::string Destroyer::getModel() const {
25     return "Super-class Star Destroyer";
26 }
27
28 unsigned int Destroyer::getModelSpeedMax() const {
29     return 40;
30 }
31
```


File - C:\Heig-vd\POA\POA_L2\src\Headers\Shuttle.hpp

```
1 #ifndef POA_L2_SHUTTLE_HPP
2 #define POA_L2_SHUTTLE_HPP
3
4 #include "CargoShip.hpp"
5
6 /**
7 * Represent a specific model of a Star Wars spaceship cargo
8 * @link CargoShip
9 * @version 1.0
10 * @author Nelson Jeanrenaud
11 * @author André Marques Nora
12 */
13 class Shuttle : public CargoShip {
14     /// Count of the number of ships created
15     static unsigned int serialNumberCounter;
16     unsigned int getNextSerialNumber() override;
17 public:
18     /**
19      * Construct a shuttle
20      * @param currentCapacity weight of the load
21      * @throw invalid_argument exception
22     */
23     explicit Shuttle(double currentCapacity);
24     /** Get max capacity of the ship in tons
25      * @return max capacity of the ship in tons
26     */
27     double getMaxCapacity() const override;
28     /**
29      * Get the weight of the ship in tons
30      * @return the weight of the ship in tons
31     */
32     double getModelWeight() const override;
33     /**
34      * Get the model of the ship
35      * @return the model of the ship
36     */
37     std::string getModel() const override;
38     /**
39      * Get the maximum speed of the ship model
40      * @return the maximum speed of the ship model
41     */
42     unsigned int getModelSpeedMax() const override;
43 };
44
45
46 #endif //POA_L2_SHUTTLE_HPP
47
```


File - C:\Heig-vd\POA\POA_L2\src\Headers\TieIn.hpp

```
1 #ifndef POA_L2_TIEIN_HPP
2 #define POA_L2_TIEIN_HPP
3
4 #include "Ship.hpp"
5 /**
6  * @brief Represent a specific model of a Star Wars spaceship
7  * @link Ship
8  * @version 1.0
9  * @author Nelson Jeanrenaud
10 * @author André Marques Nora
11 */
12 class TieIn : public Ship {
13     /// Count of the number of ships created
14     static unsigned int serialNumberCounter;
15 /**
16     * @brief Get the Next Serial Number object. Increments the
17     counter
18     *
19     * @return the next serial number
20     */
21     unsigned int getNextSerialNumber() override;
22 public:
23 /**
24     * @brief Construct a new TieIn object
25     */
26     TieIn();
27     std::string getModel() const override;
28     unsigned int getModelSpeedMax() const override;
29     double getModelWeight() const override;
30 };
31
32#endif //POA_L2_TIEIN_HPP
33
```


File - C:\Heig-vd\POA\POA_L2\src\CargoShip.cpp

```
1 #include "Headers/CargoShip.hpp"
2 #include <iomanip>
3
4 CargoShip::CargoShip(unsigned int serialNumber, double currentCapacity
5   ) : currentCapacity(currentCapacity), Ship(serialNumber) {}
6
7 double CargoShip::getCurrentCapacity() const {
8   return currentCapacity;
9 }
10 std::ostream& CargoShip::toStream(std::ostream &out) const {
11   Ship::toStream(out);
12   out << std::fixed << std::setprecision(1) << "cargo : " <<
13   currentCapacity << " tons (max : "
14   << getMaxCapacity() << ")\n";
15 }
16
```



```
1 #ifndef POA_L2_SHIP_H
2 #define POA_L2_SHIP_H
3
4 #include "iostream"
5
6 class Ship;
7
8 std::ostream& operator<<(std::ostream& out, const Ship& ship);
9
10 /**
11  * Represent a Star Wars spaceship
12  * @version 1.0
13  * @author Nelson Jeanrenaud
14  * @author André Marques Nora
15  */
16
17 class Ship {
18     /// nickname given to this spaceship, can be null
19     std::string nickname;
20     /// serial number of this ship
21     unsigned int serialNumber;
22 protected:
23     /**
24      * Construct a new Ship object
25      *
26      * @param serialNumber serial number of this ship
27      */
28     explicit Ship(unsigned int serialNumber);
29
30 public:
31     virtual ~Ship();
32     friend std::ostream& operator<<(std::ostream& out, const Ship&
33     ship);
34     virtual std::ostream& toStream(std::ostream& out) const;
35     /**
36      * Give a new nickname to this ship
37      *
38      * @param newNickname nickname given to this ship
39     void setNickname(const std::string &newNickname);
40     /**
41      * Get the nickname of the ship
42      * @return the nickname of the ship, if the ship has no nickname,
43      * returns an empty string
44     */
45     std::string getNickname() const;
46     /**
47      * Get the consumption of the ship in tons
48      * @param distance distance traveled in mio km
49      * @param speed speed of the ship in MGLT
50      * @return the consumption of the ship in tons
51     */
52     double getConsumption(unsigned distance, unsigned speed) const;
```

```
52     /**
53      * Get the Model object
54      *
55      * @return the model of the ship
56      */
57     virtual std::string getModel() const = 0;
58 /**
59      * Get the maximum speed of the ship model
60      *
61      * @return the maximum speed of the ship model
62      */
63     virtual unsigned int getModelSpeedMax() const = 0;
64 /**
65      * Get the weight of the ship model
66      *
67      * @return the weight of the ship model
68      */
69     virtual double getModelWeight() const = 0;
70 /**
71      * Get the serial number of the next ship to be built
72      *
73      * @return the serial number of the next ship to be built
74      */
75     virtual unsigned int getNextSerialNumber() = 0;
76 };
77
78
79 #endif //POA_L2_SHIP_H
80
```

File - C:\Heig-vd\POA\POA_L2\src\Headers\Squadron.hpp

```
1 #ifndef POA_L2_SQUADRON_HPP
2 #define POA_L2_SQUADRON_HPP
3
4 #include <vector>
5 #include "Ship.hpp"
6
7 /**
8 * Representation of a squadron of ships
9 * @link Ship
10 * @version 1.0
11 * @author Nelson Jeanrenaud
12 * @author André Marques Nora
13 *
14 */
15 class Squadron {
16     /**
17      * Member of the squadron, node of the linked list
18      */
19     class Member {
20         /**
21          * Pointer to the ship of the member
22          */
23         const Ship* ship;
24         /**
25          * Pointer to the next member of the squadron
26          */
27         Member* next;
28     public:
29         /**
30          * Construct a new Member object
31          * @param ship the ship of the member
32          */
33         explicit Member(const Ship* ship);
34         /**
35          * Get the ship of the member
36          * @return the ship of the member
37          */
38         const Ship& getShip() const;
39         /**
40          * Get the next member of the squadron
41          * @return the next member of the squadron
42          */
43         Member* getNext() const;
44         /**
45          * Set the next member of the squadron
46          * @param next the next member of the squadron
47          */
48         void setNext(Member* next);
49
50         /**
51          * Checks if the member has a next member
52          * @return true if the member has a next member, false otherwise
53          */
54     }
```

```

54     bool hasNext() const;
55 };
56
57     /// Ships contained in the squadron
58     Member *firstMember;
59
60     /// Leader of the squadron
61     Member *leader;
62
63     /// Name of the squadron
64     std::string name;
65
66     /**
67      * Get the member of the squadron that has the ship passed as
68      * @param parameter-ship the ship to search
69      * @return the member of the squadron that has the ship passed as
70      * parameter.
71
72     */
73     Member* getMember(const Ship& ship) const;
74     /**
75      * Get the member of the squadron at the given index
76      *
77      * @param index the index of the member to get
78      * @return Member* the member of the squadron at the given index
79      * @throw invalid_argument exception
80      */
81     Member* getMember(unsigned index) const;
82     /**
83      * Checks if the squadron is empty.
84      * @return true if the squadron is empty, false otherwise
85      */
86     bool isEmpty() const;
87     /**
88      * Empty the squadron
89      * @param squadron the squadron to copy
90      */
91     void emptySquad();
92     /**
93      * Copy a squadron
94      * @param squadron squadron to copy
95      */
96     void copySquad(const Squadron& squadron);
97 public:
98     /**
99      * Construct a new Squadron object
100     * @param name the name of the squadron
101     */
102     explicit Squadron(const std::string &name);
103     /**
104      * Construct a new Squadron object by copying the given squadron
105      */

```

```
105     * @param squadron the squadron to copy
106     */
107     Squadron(const Squadron& squadron);
108     /**
109      * operator of affectation
110      * @param squadron
111      * @return Squadron
112      */
113     Squadron& operator=(const Squadron& squadron);
114     /**
115      * Destroys the Squadron object
116      */
117     ~Squadron();
118
119     /**
120      * Writes the details of the squadron into the output stream
121      * @param out output stream
122      * @param squadron squadron to be written
123      * @return output stream
124      */
125     friend std::ostream& operator<<(std::ostream& out, const Squadron
126     & squadron);
127     /**
128      * Set leader of the squadron
129      * @param ship the ship to set as leader
130      * @throw invalid_argument exception
131      */
132     void setLeader(const Ship& ship);
133     /**
134      * Remove leader of the squadron
135      */
136     void removeLeader();
137     /**
138      * Get leader of the squadron
139      * @return the leader of the squadron
140      */
141     const Ship* getLeader() const;
142     /**
143      * Set name of squadron
144      * @param name squadron's name
145      */
146     void setName(std::string name);
147     /**
148      * Get name of squadron
149      * @return string squadron's name
150      */
151     std::string getName() const;
152     /**
153      * Writes the details of the squadron into the output stream.
154      * @param out output stream
155      */
156     std::ostream& toStream(std::ostream &out) const;
```

```
157     /**
158      * Adds a ship to the squadron
159      * @param ship the ship to add
160      * @throw invalid_argument exception
161     */
162     void addShipToSelf(const Ship& ship);
163     /**
164      * Adds a ship to a copy of the squadron
165      * @param ship the ship to add
166      * @return the copy of the squadron with the ship added
167     */
168     Squadron addShip(const Ship& ship) const;
169     /**
170      * Removes a ship from the squadron
171      * @param ship the ship to remove
172      * @throw invalid_argument exception
173     */
174     void removeShipToSelf(const Ship& ship);
175     /**
176      * Removes a ship from a copy of the squadron
177      * @param ship the ship to remove
178      * @return the copy of the squadron with the ship removed
179     */
180     Squadron removeShip(const Ship& ship) const;
181     /**
182      * Adds a ship to the squadron
183      * @param ship the ship to add
184     */
185     Squadron& operator+=(const Ship& ship);
186     /**
187      * Removes a ship from the squadron
188      * @param ship the ship to remove
189     */
190     Squadron& operator-=(const Ship& ship);
191     /**
192      * Adds a ship to a copy of the squadron
193      * @param ship the ship to add
194      * @return the copy of the squadron with the ship added
195     */
196     Squadron operator+(Ship& ship) const;
197     /**
198      * Removes a ship from a copy of the squadron
199      * @param ship the ship to remove
200      * @return the copy of the squadron with the ship removed
201     */
202     Squadron operator-(const Ship& ship) const;
203     /**
204      * Get the member of the squadron at the given index
205      *
206      * @param index the index of the member to get
207      * @return Member* the member of the squadron at the given index
208     */
209 
```

```
File - C:\Heig-vd\POA\POA_L2\src\Headers\Squadron.hpp
210     const Ship& operator[](unsigned index) const;
211
212     /**
213      * Get the size of the squadron
214      * @return the size of the squadron
215     */
216     size_t getSize() const;
217
218     /**
219      * Get the maximum speed this squadron can go at
220      *
221      * @return the maximum speed this squadron can go at
222     */
223     unsigned int getMaximumSpeed() const;
224     /**
225      * Get the total weight of this squadron
226      *
227      * @return the total weight of this squadron
228     */
229     double getTotalWeight() const;
230     /**
231      * Get the consumption of the squadron in tons
232      * @param distance distance in mio kilometers
233      * @param speed speed in MGHT
234      * @return the consumption of the squadron in tons
235      * @throw invalid_argument exception
236     */
237     double getConsumption(unsigned distance, unsigned speed) const;
238 }
239
240 #endif //POA_L2_SQUADRON_HPP
241
```


File - C:\Heig-vd\POA\POA_L2\src\Headers\CargoShip.hpp

```
1 #ifndef POA_L2_CARGOSHIP_HPP
2 #define POA_L2_CARGOSHIP_HPP
3
4 #include "Ship.hpp"
5
6 /**
7 * Represent a spaceship with a cargo
8 * @Link Ship
9 * @version 1.0
10 * @author Nelson Jeanrenaud
11 * @author André Marques Nora
12 */
13 class CargoShip : public Ship {
14     protected:
15         double currentCapacity;
16     /**
17      * @brief Construct a new Cargo Ship object
18      *
19      * @param serialNumber the serial number of the ship
20      * @param currentCapacity the current capacity of the ship
21      */
22     CargoShip(unsigned int serialNumber, double currentCapacity);
23     public:
24     /**
25      * Get max capacity of the ship in tons
26      * @return max capacity of the ship in tons
27      */
28     virtual double getMaxCapacity() const = 0;
29     /**
30      * Get the current capacity of the ship in tons
31      * @return the current capacity of the ship in tons
32      */
33     double getCurrentCapacity() const;
34
35     /**
36      * Writes the details of the squadron into the output stream.
37      * @param out output stream
38      * @return output stream
39      */
40     std::ostream& toStream(std::ostream &out) const;
41 };
42
43
44 #endif //POA_L2_CARGOSHIP_HPP
45
```


File - C:\Heig-vd\POA\POA_L2\src\TieLn.cpp

```
1 #include "Headers/TieLn.hpp"
2
3 unsigned int TieLn::serialNumberCounter = 0;
4
5 unsigned int TieLn::getNextSerialNumber() {
6     return ++serialNumberCounter;
7 }
8
9 double TieLn::getModelWeight() const {
10    return 6;
11 }
12
13 unsigned int TieLn::getModelSpeedMax() const {
14    return 100;
15 }
16
17 std::string TieLn::getModel() const {
18    return "TIE/LN";
19 }
20
21 TieLn::TieLn() : Ship(getNextSerialNumber()) {}
22
23
```


File - C:\Heig-vd\POA\POA_L2\src\Headers\Destroyer.hpp

```
1 #ifndef POA_L2_DESTROYER_HPP
2 #define POA_L2_DESTROYER_HPP
3
4 #include "CargoShip.hpp"
5
6 /**
7 * Represent a specific model of a Star Wars spaceship cargo
8 * @link CargoShip
9 * @version 1.0
10 * @author Nelson Jeanrenaud
11 * @author André Marques Nora
12 */
13 class Destroyer : public CargoShip {
14     /// Count of the number of ships created
15     static unsigned int serialNumberCounter;
16     unsigned int getNextSerialNumber() override;
17 public:
18     /**
19      * Construct a new Destroyer ship object
20      * @param currentCapacity - current capacity of the ship in tons
21      * @throw invalid_argument exception
22      */
23     explicit Destroyer(double currentCapacity);
24
25     /**
26      * Get max capacity of the ship in tons
27      * @return max capacity of the ship in tons
28      */
29     double getMaxCapacity() const override;
30
31     /**
32      * Get the weight of the ship in tons
33      * @return the weight of the ship in tons
34      */
35     double getModelWeight() const override;
36
37     /**
38      * Get the model of the ship
39      * @return the model of the ship
40      */
41     std::string getModel() const override;
42
43     /**
44      * Get the maximum speed of the ship model
45      * @return the maximum speed of the ship model
46      */
47     unsigned int getModelSpeedMax() const override;
48 };
49
50
51 #endif //POA_L2_DESTROYER_HPP
52
```


File - C:\Heig-vd\POA\POA_L2\src\Headers\TieLn.hpp

```
1 #ifndef POA_L2_TIELN_HPP
2 #define POA_L2_TIELN_HPP
3
4
5 #include "Ship.hpp"
6 /**
7 * Represent a specific model of a Star Wars spaceship
8 * @Link Ship
9 * @version 1.0
10 * @author Nelson Jeanrenaud
11 * @author André Marques Nora
12 */
13 class TieLn : public Ship {
14     /// Count of the number of ships created
15     static unsigned int serialNumberCounter;
16 /**
17     * Get the Next Serial Number object. Increments the counter
18     *
19     * @return the next serial number
20     */
21     unsigned int getNextSerialNumber() override;
22 public:
23     /**
24     * Construct a new TieLn object
25     */
26     TieLn();
27     std::string getModel() const override;
28     unsigned int getModelSpeedMax() const override;
29     double getModelWeight() const override;
30 };
31
32
33 #endif //POA_L2_TIELN_HPP
34
```



```
1 #include "Headers/Ship.hpp"
2 #include <cmath>
3 #include <iomanip>
4
5 Ship::Ship(unsigned int serialNumber) : nickname(""), serialNumber(
6     serialNumber) {}
7
8 void Ship::setNickname(const std::string &newNickname) {
9     if (newNickname.empty()) {
10         nickname = "";
11     } else {
12         nickname = newNickname;
13     }
14 }
15 double Ship::getConsumption(unsigned distance, unsigned speed) const {
16     if(speed > this->getModelSpeedMax() || speed <= 0)
17         throw std::invalid_argument("Speed must be greater than 0 and
18             less than or equal to max speed");
19     return std::cbrt(getModelWeight()) / 2 * log10(getModelWeight() *
20         speed) * log10(distance + 1);
21 }
22 std::ostream& Ship::toStream(std::ostream &out) const {
23     out << std::fixed << std::setprecision(2) << (nickname.empty() ?
24         "" : nickname + " ");
25     out << "[" << getModel() << " #" << serialNumber << "]\\n"
26     << " weight : " << getModelWeight() << " tons\\n"
27     << " max speed : " << getModelSpeedMax() << " MGLT\\n";
28     return out;
29 }
30
31 std::string Ship::getNickname() const {
32     return this->nickname;
33 }
34
35 Ship::~Ship() {
36 }
37
38 }
39
```



```
1 #include "Headers/Shuttle.hpp"
2
3 unsigned int Shuttle::serialNumberCounter = 0;
4
5 unsigned int Shuttle::getNextSerialNumber() {
6     return ++serialNumberCounter;
7 }
8
9 Shuttle::Shuttle(double currentCapacity) : CargoShip(
10    getNextSerialNumber(), currentCapacity) {
11    if(currentCapacity < 0 || currentCapacity > this->getMaxCapacity()
12       ()) {
13        throw std::invalid_argument("Capacity must be greater than 0 and
14 less than or equal to max capacity");
15    }
16 }
17
18
19 double Shuttle::getMaxCapacity() const {
20     return 80.0;
21 }
22
23 std::string Shuttle::getModel() const {
24     return "Lambda-class shuttle";
25 }
26
27 unsigned int Shuttle::getModelSpeedMax() const {
28     return 54;
29 }
```



```
1 #include "Headers/Squadron.hpp"
2
3 #include <utility>
4
5 Squadron::Squadron(const std::string &name) : name(name), firstMember(
6     nullptr), leader(nullptr) {
7 }
8
9 Squadron::Squadron(const Squadron &squadron) : Squadron::Squadron(
10    squadron.name) {
11     /* Copying the squadron. */
12     this->copySquad(squadron);
13 }
14
15 Squadron::~Squadron() {
16     /* Deleting the members of the squadron. */
17     Member *currentMember = firstMember;
18     while (currentMember != nullptr) {
19         Member *nextMember = currentMember->getNext();
20         delete currentMember;
21         currentMember = nextMember;
22     }
23 }
24
25 Squadron &Squadron::operator=(const Squadron &squadron) {
26     this->emptySquad();
27
28     this->copySquad(squadron);
29 }
30
31 void Squadron::setLeader(const Ship& ship) {
32     Member *member = getMember(ship);
33     if (member == nullptr) {
34         throw std::invalid_argument("Ship is not in the squadron");
35     }
36     leader = member;
37 }
38
39 bool Squadron::isEmpty() const {
40     return firstMember == nullptr;
41 }
42
43 Squadron::Member* Squadron::getMember(const Ship &ship) const{
44     if(isEmpty()) {
45         return nullptr;
46     }
47     Member *currentMember = firstMember;
48     while (currentMember != nullptr) {
49         if (&currentMember->getShip() == &ship) {
50             return currentMember;
51         }
52     }
53 }
```

```
52         currentMember = currentMember->getNext();
53     }
54     return nullptr;
55 }
56
57 Squadron::Member* Squadron::getMember(unsigned index) const{
58     size_t size = this->getSize() - 1;
59     if(index < 0 || index > size)
60         throw std::invalid_argument("Index out of bound");
61
62     Member* m = this->firstMember;
63     for (int i = 0; i < size - index; ++i) {
64         if (m->hasNext())
65             m = m->getNext();
66     }
67     return m;
68 }
69
70 void Squadron::addShipToSelf(const Ship &ship) {
71     if(isEmpty()) {
72         firstMember = new Member(&ship);
73     } else if(getMember(ship) != nullptr){
74         throw std::invalid_argument("Ship is already in squadron");
75     }
76     else {
77         Member* oldFirstMember = firstMember;
78         firstMember = new Member(&ship);
79         firstMember->setNext(oldFirstMember);
80     }
81 }
82
83 Squadron Squadron::addShip(const Ship& ship) const{
84     Squadron newSquadron(*this);
85     newSquadron.addShipToSelf(ship);
86     return newSquadron;
87 }
88
89 void Squadron::removeShipToSelf(const Ship &ship) {
90     if(isEmpty()) {
91         throw std::invalid_argument("Squadron is empty");
92     }
93     Member *currentMember = firstMember;
94     Member *previousMember = nullptr;
95     while (currentMember != nullptr) {
96         if (&currentMember->getShip() == &ship) {
97             if (previousMember == nullptr) {
98                 firstMember = currentMember->getNext();
99             } else {
100                 previousMember->setNext(currentMember->getNext());
101             }
102             if(currentMember == leader)
103                 leader = nullptr;
104 }
```

```

File - C:\Heig-vd\POA\POA_L2\src\Squadron.cpp
105         delete currentMember;
106         return;
107     }
108     previousMember = currentMember;
109     currentMember = currentMember->getNext();
110 }
111 throw std::invalid_argument("Ship is not in the squadron");
112 }
113
114 Squadron Squadron::removeShip(const Ship &ship) const{
115     Squadron newSquadron(*this);
116     newSquadron.removeShipToSelf(ship);
117     return newSquadron;
118 }
119
120 unsigned int Squadron::getMaximumSpeed() const {
121     if(isEmpty())
122         return 0;
123
124     unsigned int maxSpeed = firstMember->getShip().getModelSpeedMax
125     ();
126     Member *currentMember = firstMember->getNext();
127     while (currentMember != nullptr) {
128         if (currentMember->getShip().getModelSpeedMax() < maxSpeed) {
129             maxSpeed = currentMember->getShip().getModelSpeedMax();
130         }
131         currentMember = currentMember->getNext();
132     }
133     return maxSpeed;
134 }
135 double Squadron::getTotalWeight() const {
136     if(isEmpty()) {
137         return 0;
138     }
139     double totalWeight = 0;
140     Member *currentMember = firstMember;
141     while (currentMember != nullptr) {
142         totalWeight += currentMember->getShip().getModelWeight();
143         currentMember = currentMember->getNext();
144     }
145     return totalWeight;
146 }
147
148 double Squadron::getConsumption(unsigned distance, unsigned speed)
149 const {
150     if(isEmpty()) {
151         return 0;
152     }
153     if(speed > getMaximumSpeed()) {
154         throw std::invalid_argument("Speed is greater than this
squadron's maximum speed");
155     }

```

```
155     double consumption = 0;
156     Member *currentMember = firstMember;
157     while (currentMember != nullptr) {
158         consumption += currentMember->getShip().getConsumption(
159             distance, speed);
160         currentMember = currentMember->getNext();
161     }
162     return consumption;
163 }
164 std::ostream& Squadron::toStream(std::ostream &out) const {
165     out << "Squadron: " << name << " :\n";
166     out << " max speed: " << getMaximumSpeed() << " MGLT\n";
167     out << " total weight: " << getTotalWeight() << " tons\n";
168     out << "-- Leader:\n";
169     if(leader != nullptr) {
170         out << this->getLeader();
171     } else {
172         out << "None\n";
173     }
174     out << "\n-- Members:\n";
175     if(firstMember != nullptr) {
176         Squadron::Member *currentMember = firstMember;
177         while (currentMember != nullptr) {
178             if (currentMember != leader) {
179                 out << "\n" << currentMember->getShip() << "\n";
180             }
181             currentMember = currentMember->getNext();
182         }
183     } else {
184         out << "None\n";
185     }
186 }
187     return out;
188 }
189
190 Squadron& Squadron::operator+=(const Ship& ship) {
191     addShipToSelf(ship);
192     return *this;
193 }
194
195 Squadron& Squadron::operator-=(const Ship &ship) {
196     removeShipToSelf(ship);
197     return *this;
198 }
199
200 /// Member
201
202 Squadron::Member::Member(const Ship *ship) : ship(ship), next(nullptr)
203     {}
204 Squadron::Member* Squadron::Member::getNext() const{
205     return next;
```

```
File - C:\Heig-vd\POA\POA_L2\src\Squadron.cpp
206 }
207
208 const Ship& Squadron::Member::getShip() const{
209     return *ship;
210 }
211
212 void Squadron::Member::setNext(Squadron::Member *next) {
213     this->next = next;
214 }
215
216 bool Squadron::Member::hasNext() const{
217     return next != nullptr;
218 }
219
220 /// Friend
221 std::ostream& operator<<(std::ostream& out, const Squadron& squadron
222 ) {
223     return squadron.toStream(out);
224 }
225 Squadron Squadron::operator+(Ship &ship) const{
226     return addShip(ship);
227 }
228
229 Squadron Squadron::operator-(const Ship &ship) const{
230     return removeShip(ship);
231 }
232
233 const Ship &Squadron::operator[](unsigned index) const{
234     return getMember(index)->getShip();
235 }
236
237 const Ship *Squadron::getLeader() const {
238     if(this->leader == nullptr)
239         return nullptr;
240
241     return &this->leader->getShip();
242 }
243
244 void Squadron::removeLeader() {
245     this->leader = nullptr;
246 }
247
248 size_t Squadron::getSize() const{
249     if(this->isEmpty())
250         return 0;
251
252     Member* m = this->firstMember;
253     unsigned int counter = 1;
254     while (m->hasNext()){
255         counter++;
256         m = m->getNext();
257     }
```

File - C:\Heig-vd\POA\POA_L2\src\Squadron.cpp

```
258     return counter;
259 }
260
261 void Squadron::emptySquad() {
262     for (int i = 0; i < this->getSize(); ++i) {
263         this->removeShip(this->getMember(i)->getShip());
264     }
265     this->removeLeader();
266 }
267
268 void Squadron::setName(std::string name) {
269     this->name = std::move(name);
270 }
271
272 std::string Squadron::getName() const{
273     return this->name;
274 }
275
276 void Squadron::copySquad(const Squadron &squadron) {
277     Member *currentMember = squadron.firstMember;
278     while (currentMember != nullptr) {
279         addShipToSelf(currentMember->getShip());
280         currentMember = currentMember->getNext();
281     }
282     if(squadron.leader != nullptr)
283         setLeader(squadron.leader->getShip());
284 }
285
```

```
1 #include <iostream>
2 #include "Headers/Squadron.hpp"
3 #include "Headers/TieLn.hpp"
4 #include "Headers/Shuttle.hpp"
5 #include "Headers/TieIn.hpp"
6 #include "Headers/Destroyer.hpp"
7
8 int main(){
9
10    /*
11     * Test class Ship
12     */
13
14    TieLn blackLeader;
15    const TieIn blackOne;
16    Destroyer blackTwo(320.6);
17    Shuttle blackThree(23.4);
18    //Shuttle overWeight(90.4);
19
20    blackLeader.setNickname("Black Star");
21
22    std::cout << blackLeader << std::endl;
23    std::cout << blackTwo << std::endl;
24
25    std::cout << "Consumption of blackOne for a distance of 1000 and
26 speed at 50 MGLT : "
27           << blackOne.getConsumption(1000,50) << std::endl;
28
29    //std::cout << "Consumption of blackTwo for a distance of 1000 and
30 speed at 1000 MGLT : "
31    //           << blackTwo.getConsumption(1000,1000) << std::endl;
32
33    std::cout << "Consumption of blackThree for a distance of 1000 and
34 speed at 50 MGLT : "
35           << blackThree.getConsumption(1000,50) << std::endl;
36
37    std::cout << "Consumption of blackThree for a distance of -1000 and
38 speed at 50 MGLT : "
39           << blackThree.getConsumption(-1000,50) << std::endl;
40
41    std::cout << "Consumption of blackThree for a distance of 1000 and
42 speed at -50 MGLT : "
43           << blackThree.getConsumption(1000,-50) << std::endl;
44
45    std::cout << "Consumption of blackThree for a distance of 1000 and
46 speed at 500 MGLT : "
47           << blackThree.getConsumption(1000,500) << std::endl;
48 */
49
50    std::cout << std::endl;
51
52    /*
53     * Test class Squadron
54     */
```

```
48     */
49
50 // constructor, add ship to squad and set a leader
51 Squadron blackSquad("Black Squadron");
52
53 std::cout << blackSquad.getName() << std::endl;
54
55 std::cout << blackSquad << std::endl;
56
57 blackSquad += blackLeader;
58 blackSquad += blackOne;
59 blackSquad + blackTwo;
60 blackSquad += blackThree;
61
62 //blackSquad += blackThree;
63 //blackSquad + blackThree;
64
65 blackSquad.setLeader(blackLeader);
66
67 std::cout << blackSquad << std::endl;
68
69 std::cout << "Consumption of Black Squad for a distance of 1000
    and speed at 30 MGLT : "
70     << blackSquad.getConsumption(1000,30) << std::endl;
71 //std::cout << blackSquad.getConsumption(1000,300) << std::endl;
72
73 std::cout << std::endl;
74
75 //constructor by copy, remove ship from squad and remove leader
76 Squadron blackSquad2(blackSquad);
77
78 std::cout << blackSquad2 << std::endl;
79
80 blackSquad2.removeLeader();
81
82 std::cout << "squad2" << blackSquad2 << std::endl;
83
84 blackSquad2 -= blackLeader;
85 blackSquad2 - blackThree;
86
87 //blackSquad2 -= blackLeader;
88 //blackSquad2 - blackLeader;
89
90 std::cout << blackSquad2 << std::endl;
91
92 TieLn blackLeader2;
93 blackLeader2.setNickname("Black Comet");
94
95 //blackSquad2.setLeader(&blackLeader2);
96
97 blackSquad2 += blackLeader2;
98 blackSquad2.setLeader(blackLeader2);
99
```

```
100    std::cout << blackSquad2 << std::endl;
101
102 //operator= and operator[]
103 std::cout << "operator = & []" << std::endl;
104 std::cout << std::endl;
105
106 Destroyer blueLeader(60);
107 Shuttle blueOne(10);
108
109 Squadron blueSquad("Blue Squadron");
110
111 blueSquad.removeLeader();
112
113 blueSquad += blueLeader;
114
115 //blueSquad.setLeader(&blueOne);
116
117 blueSquad += blueOne;
118
119 blueSquad.setLeader(blueLeader);
120 std::cout << blueSquad << std::endl;
121
122 blueSquad -= blueLeader;
123
124 std::cout << "leader " << (blueSquad.getLeader() == nullptr ? "
test good" : " test failed") << std::endl;
125
126 blueSquad = blackSquad;
127
128 blueSquad.setName("Blue Monkey");
129
130 std::cout << blueSquad << std::endl;
131
132 std::cout << blueSquad[0] << std::endl;
133
134 /*std::cout << blueSquad[10] << std::endl;
135
136 std::cout << "Consumption of blacksquad for a distance of 1000
and speed at 50 MGLT : "
137             << blackSquad.getConsumption(1000,50) << std::endl;
138
139 /* std::cout << "Consumption of blacksquad for a distance of -1000
and speed at 50 MGLT : "
140             << blackSquad.getConsumption(-1000,50) << std::endl;
141
142 std::cout << "Consumption of blacksquad for a distance of 1000
and speed at -50 MGLT : "
143             << blackSquad.getConsumption(1000,-50) << std::endl;
144
145 std::cout << "Consumption of blacksquad for a distance of 1000
and speed at 500 MGLT : "
146             << blackSquad.getConsumption(1000,500) << std::endl;
147 */
```

File - C:\Heig-vd\POA\POA_L2\src\Test.cpp

```
148 const Squadron constantSquad("constant");
149 const Squadron copyConstant(constantSquad);
150 const Squadron copyConstant2(blueSquad);
151 Squadron copyConstant3(constantSquad);
152
153 const Squadron constantSquad2 = constantSquad;
154 const Squadron constantSquad3(constantSquad + blackThree);
155 const Squadron constantSquad4(blueSquad + blackTwo);
156
157
158 return 0;
159 }
160
```

File - C:\Heig-vd\POA\POA_L2\src\TieIn.cpp

```
1 #include "Headers/TieIn.hpp"
2
3 unsigned int TieIn::serialNumberCounter = 0;
4
5 unsigned int TieIn::getNextSerialNumber() {
6     return ++serialNumberCounter;
7 }
8
9 double TieIn::getModelWeight() const {
10    return 5;
11 }
12
13 unsigned int TieIn::getModelSpeedMax() const {
14    return 110;
15 }
16
17 std::string TieIn::getModel() const {
18    return "TIE/IN";
19 }
20
21 TieIn::TieIn() : Ship(getNextSerialNumber()) {}
22
```