```cpp
#include "Controller.h"
#include <algorithm>
#include <vector>
#include <iterator>

/// ---------- Repository ---------- ///

void Controller::addCoatToRepository(const std::string & ID, const int & size, const std::string & colour, const double & price, const int & quantity, const std::string & link)
{
    Coat coat{ ID, size, colour, price, quantity, link };
    this->repo.addCoat(coat);
}

void Controller::removeCoatFromRepository(const std::string & ID)
{
    this->repo.removeCoatByID(ID);
}

void Controller::updateCoatToRepository(const std::string & ID, const double & new_price, const int & new_quantity, const std::string & new_link)
{
    Coat existingCoat = this->repo.findByID(ID);
    Coat newCoat{ ID, existingCoat.getSize(), existingCoat.getColour(), new_price, new_quantity, new_link };
    this->repo.updateCoat(ID, newCoat);
}

/// ---------- Shopping Cart ---------- ///

void Controller::clearProducts()
{
    this->cart.clearProducts();
}

void Controller::addAllAvailableCoats()
{
    // C++11 method
    //std::for_each(this->getAllCoats().begin(), this->getAllCoats().end(), this->cart.addAvailableCoats);
    for (auto&& coat : this->getAllCoats()) {
        this->cart.addAvailableCoats(coat);
    }

    /* Classic method
    std::vector<Coat> coats = getAllCoats();

    for (int i = 0; i < coats.size(); i++) {
        this->cart.addAvailableCoats(coats[i]);
    }
    */
}

void Controller::addAllSizeCoats(const int & size)
{
    // C++11 method
    std::vector<Coat> coats_source = this->getAllCoats();
    std::vector<Coat> coats_dest;
    std::copy_if(coats_source.begin(), coats_source.end(), std::back_inserter(coats_dest), [&](Coat c) {return c.getSize() == size; });
```

```
56      for (auto&& coat : coats_dest) {
57          this->cart.addAvailableCoats(coat);
58      }
59
60      // Classic method
61      /*std::vector<Coat> coats = getAllCoats();
62      for (int i = 0; i < coats.size(); i++) {
63          if (coats[i].getSize() == size)
64              this->cart.addAvailableCoats(coats[i]);
65      }*/
66
67  }
68
69  void Controller::addCoatToCart(const Coat & c)
70  {
71      this->cart.add(c);
72  }
73
74  void Controller::startShopping()
75  {
76      this->cart.start();
77  }
78
79  void Controller::nextCoatShopping()
80  {
81      this->cart.next();
82  }
83
84  void Controller::buyProducts()
85  {
86      std::vector<Coat> coatsInCart = this->cart.getCartContents();
87      for (int i = 0; i < coatsInCart.size(); i++) {
88          this->repo.sellCoatByID(coatsInCart[i].getID());
89      }
90      this->eraseCart();
91      this->cart.clearProducts();
92  }
93
94  void Controller::eraseCart()
95  {
96      this->cart.clearCart();
97  }
98
```