Miko Farin (704789591)

CS31

Project 5 Report

1.Notable obstacles I went through are the way I setup the getkind() constructor. I had trouble with this piece of code and delayed my project. The trouble I was having was making sure that the way I obtained the mKind was correct an I did not use the enumeration at first to be able to make the constructor into a Kind. Another obstacle I face was learning how to use namespaces and creating my own namespace. I figured this out by learning that you should wrap all your files in a namespace and name that namespace. Lastly, I had trouble knowing that you can call AppleProduct into the AppleStore.cpp file and use that to return the values you want to create.

2.List of Test Data:

1.

cs31::AppleProduct p;

assert(p.getCost() == 0.0);

assert(p.getKind() == cs31::AppleProduct::Kind::NOTSPECIFIED);

assert(p.getColor() == "");

assert(p.getVersion() == "");

I used this test case to see what would the object have without making any variables or definitions to the object. Provided by the SPEC.

2. cs31::AppleProduct p1(cs31::AppleProduct::Kind::IPHONE, "8", "Space Grey", 699.00);

assert(std::to\_string(p1.getCost()) == "699.000000");

assert(p1.getKind() == cs31::AppleProduct::Kind::IPHONE);

assert(p1.getColor() == "Space Grey");

assert(p1.getVersion() == "8");

I used this test case to see what would the object have when making manipulations to the objects properties, giving it variables to pass to the constructor. Also test the way it would store the proper information to be able to be accessed later in AppleStore.

3. cs31::AppleProduct p2(cs31::AppleProduct::Kind::IPAD, "Pro", "Silver", 649.00);

assert(std::to\_string(p2.getCost()) == "649.000000");

assert(p2.getKind() == cs31::AppleProduct::Kind::IPAD);

assert(p2.getColor() == "Silver");

assert(p2.getVersion() == "Pro");

I used this test case to see what would the object have when making manipulations to the objects properties, giving it variables to pass to the constructor. Also test the way it would store the proper information to be able to be accessed later in AppleStore.

4. cs31::AppleProduct p3(cs31::AppleProduct::Kind::WATCH, "1", "BLACK", 249.00);

assert(std::to\_string(p3.getCost()) == "249.000000");

assert(p3.getKind() == cs31::AppleProduct::Kind::WATCH);

assert(p3.getColor() == "BLACK");

assert(p3.getVersion() == "1");

I used this test case to see what would the object have when making manipulations to the objects properties, giving it variables to pass to the constructor. Also test the way it would store the proper information to be able to be accessed later in AppleStore.

5. cs31::AppleProduct p4(cs31::AppleProduct::Kind::IPHONE, "X", "BLACK", 999.00);

assert(std::to\_string(p4.getCost()) == "999.000000");

assert(p4.getKind() == cs31::AppleProduct::Kind::IPHONE);

assert(p4.getColor() == "BLACK");

assert(p4.getVersion() == "X");

I used this test case to see what would the object have when making manipulations to the objects properties, giving it variables to pass to the constructor. Also test the way it would store the proper information to be able to be accessed later in AppleStore.

6. cs31::AppleStore store;

cs31::AppleProduct iphone8\_64 = store.buyiPhone8("Space Grey", 64);

assert(std::to\_string(iphone8\_64.getCost()) == "699.000000");

assert(iphone8\_64.getKind() == cs31::AppleProduct::Kind::IPHONE);

assert(iphone8\_64.getColor() == "Space Grey");

assert(iphone8\_64.getVersion() == "8");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

7. cs31::AppleStore store1;

cs31::AppleProduct buywatch\_1 = store1.buyWatch1("White");

assert(std::to\_string(buywatch\_1.getCost()) == "249.000000");

assert(buywatch\_1.getKind() == cs31::AppleProduct::Kind::WATCH);

assert(buywatch\_1.getColor() == "White");

assert(buywatch\_1.getVersion() == "1");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

8. cs31::AppleStore store2;

cs31::AppleProduct buywatch\_3 = store2.buyWatch3("BLUE");

assert(std::to\_string(buywatch\_3.getCost()) == "329.000000");

assert(buywatch\_3.getKind() == cs31::AppleProduct::Kind::WATCH);

assert(buywatch\_3.getColor() == "BLUE");

assert(buywatch\_3.getVersion() == "3");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

9. cs31::AppleStore store3;

cs31::AppleProduct ipad = store.buyiPad("Green", 32);

assert(std::to\_string(ipad.getCost()) == "329.000000");

assert(ipad.getKind() == cs31::AppleProduct::Kind::IPAD);

assert(ipad.getColor() == "Green");

assert(ipad.getVersion() == "");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

10. cs31::AppleStore store4;

cs31::AppleProduct ipad\_2 = store.buyiPad("Purple", 128);

assert(std::to\_string(ipad\_2.getCost()) == "429.000000");

assert(ipad\_2.getKind() == cs31::AppleProduct::Kind::IPAD);

assert(ipad\_2.getColor() == "Purple");

assert(ipad\_2.getVersion() == "");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

11. cs31::AppleStore store5;

cs31::AppleProduct ipad\_pro = store.buyiPadPro("Gold", 64);

assert(std::to\_string(ipad\_pro.getCost()) == "649.000000");

assert(ipad\_pro.getKind() == cs31::AppleProduct::Kind::IPAD);

assert(ipad\_pro.getColor() == "Gold");

assert(ipad\_pro.getVersion() == "Pro");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

12. cs31::AppleStore store6;

cs31::AppleProduct ipad\_pro\_256 = store.buyiPadPro("LOL", 256);

assert(std::to\_string(ipad\_pro\_256.getCost()) == "799.000000");

assert(ipad\_pro\_256.getKind() == cs31::AppleProduct::Kind::IPAD);

assert(ipad\_pro\_256.getColor() == "LOL");

assert(ipad\_pro\_256.getVersion() == "Pro");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

13.

cs31::AppleStore store7;

cs31::AppleProduct ipad\_pro\_512 = store.buyiPadPro("GOLDEN BROWN", 512);

assert(std::to\_string(ipad\_pro\_512.getCost()) == "999.000000");

assert(ipad\_pro\_512.getKind() == cs31::AppleProduct::Kind::IPAD);

assert(ipad\_pro\_512.getColor() == "GOLDEN BROWN");

assert(ipad\_pro\_512.getVersion() == "Pro");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

14. cs31::AppleStore store8;

cs31::AppleProduct iPhone\_8 = store.buyiPhone8("Green", 64);

assert(std::to\_string(iPhone\_8.getCost()) == "699.000000");

assert(iPhone\_8.getKind() == cs31::AppleProduct::Kind::IPHONE);

assert(iPhone\_8.getColor() == "Green");

assert(iPhone\_8.getVersion() == "8");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

15. cs31::AppleStore store9;

cs31::AppleProduct iPhone\_8\_2 = store.buyiPhone8("Red", 256);

assert(std::to\_string(iPhone\_8\_2.getCost()) == "849.000000");

assert(iPhone\_8\_2.getKind() == cs31::AppleProduct::Kind::IPHONE);

assert(iPhone\_8\_2.getColor() == "Red");

assert(iPhone\_8\_2.getVersion() == "8");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

16. cs31::AppleStore store10;

cs31::AppleProduct iPhone\_8\_plus = store.buyiPhone8Plus("Purple", 64);

assert(std::to\_string(iPhone\_8\_plus.getCost()) == "799.000000");

assert(iPhone\_8\_plus.getKind() == cs31::AppleProduct::Kind::IPHONE);

assert(iPhone\_8\_plus.getColor() == "Purple");

assert(iPhone\_8\_plus.getVersion() == "8Plus");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

17. cs31::AppleStore store11;

cs31::AppleProduct iPhone\_8\_plus\_256 = store.buyiPhone8Plus("Silver M", 256);

assert(std::to\_string(iPhone\_8\_plus\_256.getCost()) == "949.000000");

assert(iPhone\_8\_plus\_256.getKind() == cs31::AppleProduct::Kind::IPHONE);

assert(iPhone\_8\_plus\_256.getColor() == "Silver M");

assert(iPhone\_8\_plus\_256.getVersion() == "8Plus");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

18. cs31::AppleStore store12;

cs31::AppleProduct iPhone\_X = store.buyiPhoneX("Grey", 64);

assert(std::to\_string(iPhone\_X.getCost()) == "999.000000");

assert(iPhone\_X.getColor() == "Grey");

assert(iPhone\_X.getKind() == cs31::AppleProduct::Kind::IPHONE);

assert(iPhone\_X.getVersion() == "X");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

19. cs31::AppleStore store13;

cs31::AppleProduct iPhone\_X\_256 = store.buyiPhoneX("Space Black", 256);

assert(std::to\_string(iPhone\_X\_256.getCost()) == "1149.000000");

assert(iPhone\_X\_256.getColor() == "Space Black");

assert(iPhone\_X\_256.getKind() == cs31::AppleProduct::Kind::IPHONE);

assert(iPhone\_X\_256.getVersion() == "X");

I used this code to test the functionality of Applestore. I was able to use both of the classes to retrieve the correct Apple Product and also I was able to to check if the constructor being passing to Apple Product was correct.

20.

try

{

cs31::AppleStore s;

cs31::AppleProduct iphone8 = s.buyiPhone8("Space Grey", 9999);

assert(false);

cs31::AppleStore s1;

cs31::AppleProduct iphone8\_64 = s1.buyiPhone8("Space Grey", 10000 );

assert(false);

cs31::AppleStore s3;

cs31::AppleProduct iPad = s3.buyiPad("Space Grey", 64);

assert(false);

}

catch (std::logic\_error)

{

assert(true);

}

This code was to test if my parameter for cathcing logic errors was doing its job when you input something that is not a valid int of gigabytes.