----------------------------------------------------------------------------------------------------------------------------

REFLECTION

Name: Hashmeet Singh Saini

Seneca ID : hsaini28

Student No. : 153070214

Date : November 3rd, 2022

I have done all the coding by myself and only copied the code

that my professor provided to complete my project milestones.

----------------------------------------------------------------------------------------------------------------------------

In this week’s workshop, I learnt various things like the use of STL containers (vector) and using dynamically allocated objects in this vector. Also learned the use of iterators with these containers. I strengthened my concept of reporting and handling an exception as well.

1. My vector containing different vehicles resides in the Autoshop class. The vector m\_vehicles contains pointers to different vehicles to take care of the dynamic type of different vehicle objects. Since every vehicle is dynamically allocated and then added to the vector, it is important that there is a destructor that iterates through the vector and deletes every single vehicle to avoid any memory leaks.
2. The sole purpose of loadData() function is to correctly open the file and supply the stream to the createInstnace function so that according to the record, a correct type of vehicle object can be created. But during this whole process, several errors can occur like the reading of invalid data (letters instead of numbers in file) or presence of unknown types in the records. My program handles all of these problems through exception handling during the reading process. The createInstance() function that decides what kind of vehicle is to be constructed throws and error, if the vehicle tag in the file is unknown. Similarly in the individual constructors of Car and Van, errors are thrown if there is any of the switch condition do’t match with given cases or letters are read instead of numbers when reading topSpeed. All of these thrown errors are handled and duly displayed by the loadData() function. The other way I could do this instead of throw and except was to use bool type flags which turn true if an error is detected. So, I could change the create instance function to also receive a reference of type bool so it would like this Vehicle\* createInstance(std::istream& in, bool& flag). Now when an error is detected where the record tag doesn’t match any valid tag, instead of throwing an error we could make the flag true. In the loadData() function we would then change the try-catch statements to if statements which would check that if the flag is true, that means the error has occurred. It line 31to 36 could look like this:

bool flag=false;

aVehicle = sdds::createInstance(file);

if (bool){

std::cout<<”an error with the record tag has occurred.”<<std::endl;

}

The similar flag return method could be used for the other errors too.

1. In simple words, we have used a vector of pointers instead of a vector of objects of type Vehicle because of the polymorphic design of our classes. The whole hierarchy of the classes begins from a class called Vehicle which is parent to classes Car and Van. Further Racecar and Luxuryvan have been derived from Car and Van respectively. We know that in Polymorphism, dynamic type is important because it can change. If I want to create a vector of type sdds::Vehicle then we won’t be able to store objects of type Car, Van, Racecar and Luxuryvan in the same vector. But when we create a vector of type sdds::Vehicle\*, we can add data of type all the four child classes in the same vector by dynamically allocating those objects after the determination of the tag (type of vehicle) in the createInstance function of the Utilities module.