# Assignment Description

You are an IT Manager for a small company and must create a program to track your company’s personal computer inventory.

Create a class that can be used for a Personal Computer. The class should have attributes for the:

* Manufacturer (e.g. Dell, Gateway, etc.),
* Form Factor (laptop/desktop),
* Serial Number
* Processor ( I3, I5, I7, AMD Ryzen 3, AMD Ryzen 5, etc.),
* RAM (4, 6, 8, 16, 32, or 64GB),
* Storage Type (UFS, SDD, HDD) and
* Storage Size (128GB, 256GB, 512GB, 1TB, 2TB).

The constructor must accept the manufacturer, form factor, serial number, processor, RAM, storage type/size.

Create accessor methods that allow these attributes to be retrieved individually.

Create mutator methods that allow the RAM and the storage drive (type and size) to be changed.

Incorporate exception handling to reject invalid values in the constructor and mutator methods.

Create a toString() method formulate a string containing the manufacturer, form factor, serial number, processor, RAM, and storage type/size.

Write a main program that creates a vector that can contain personal computers. The program should prompt the user for an indeterminate number of personal computers, create a personal computer object, and add the object to the vector. After each personal computer is entered the program should display the object that was just created and the total number of personal computers in the list/vector. When the user has finished entering personal computers into their inventory, the program should display the contents of the inventory list (complete list of personal computers).

# 1 Readme Documentation

This program will prompt the user for input for a computer’s manufacturer, form factor, serial number, processor, RAM, and storage size/type. Then, the user can choose to create as many computers as they wish, receiving a list of the computers they’ve made at the end of the program.

# 4 Source Code of All files (.h, .cpp)

#include *<iostream>*

#include *<iomanip>*

#include *<string>*

#include *<cctype>*

#include *<stdexcept>*

#include *<vector>*

**using** **namespace** **std**;

*/\**

*Program Name: Computer Inventory*

*Author: Wesley Hixon*

*Date Last Updated: 11/10/2024*

*Purpose: This program will create an inventory of personal computers.*

*Simply input manufacturer, form factor, serial number, processor, ram, and storage type/size*

*for as many computers as you'd like.*

*\*/*

*// Function template to get input for any data type*

**template** <**typename** **T**>

T getInput(string prompt, string errorMessage){

bool valid = false;

T input;

cout << endl << prompt;

**while**(!valid){

cin >> input;

**if**(cin.fail()){

cin.clear();

cin.ignore(10000, '\n');

**throw**(runtime\_error(errorMessage));

}

valid = true;

}

**return** input;

}

*// Enum for different storage types*

**enum** **storageTypes**{SSD, HDD, UFS};

*// Computer class to store characteristics and methods*

**class** **computer**{

**private**:

string manufacturer;

string formFactor;

int serialNum;

string processor;

int ram;

storageTypes storageType;

string storageSize;

**public**:

*// Constructor Method*

computer(

string inputManufacturer,

string inputFormFactor,

int inputSerialNum,

string inputProcessor,

int inputRam,

storageTypes inputStorageType,

string inputStorageSize

){

**try**{

setManufacturer(inputManufacturer);

setFormFactor(inputFormFactor);

setSerialNum(inputSerialNum);

setProcessor(inputProcessor);

setRam(inputRam);

setStorageType(inputStorageType);

setStorageSize(inputStorageSize);

}**catch**(**const** exception& e){

cerr << endl << e.what() << endl;

}

}

*// Default constructor initializes with no defaults*

computer(){}

*// Mutator Methods w/ error handling*

void setManufacturer(string inputManufacturer){manufacturer = inputManufacturer;}

void setFormFactor(string inputFormFactor){formFactor = inputFormFactor;}

void setSerialNum(int inputSerialNum){

**if**(inputSerialNum < 1) **throw**(invalid\_argument("Please enter a valid serial number greater than 0."));

serialNum = inputSerialNum;

}

void setProcessor(string inputProcessor){processor = inputProcessor;}

void setRam(int inputRam){

int validRam[6] = {4, 6, 8, 16, 32, 64};

bool valid = false;

*// Check if ram is valid before setting*

**for**(int i = 0; i < 6; i++){

**if**(inputRam == validRam[i]){

valid = true;

}

}

**if**(!valid) **throw**(invalid\_argument("Please enter 4, 6, 8, 16, 32, or 64 as your ram."));

ram = inputRam;

}

void setStorageType(storageTypes inputStorageType){

storageType = inputStorageType;

}

void setStorageSize(string inputStorageSize){

storageSize = inputStorageSize;

}

*// Accessor Methods*

string getManufacturer(){**return** manufacturer;}

string getFormFactor(){**return** formFactor;}

int getSerialNum(){**return** serialNum;}

string getProcessor(){**return** processor;}

int getRam(){**return** ram;}

storageTypes getStorageType(){**return** storageType;}

string getStorageSize(){**return** storageSize;}

*// Creates a string describing computer*

string toString(){

string computerString;

string storageTypeString;

**switch**(storageType){

**case** SSD:

storageTypeString = "SSD";

**break**;

**case** HDD:

storageTypeString = "HDD";

**break**;

**case** UFS:

storageTypeString = "UFS";

**break**;

}

computerString = "Manufacturer: " + manufacturer + '\n'

+ "Form Factor: " + formFactor + '\n'

+ "Serial Number: " + to\_string(serialNum) + '\n'

+ "Processor: " + processor + '\n'

+ "Ram: " + to\_string(ram) + "GB" + '\n'

+ "Storage Type: " + storageTypeString + '\n'

+ "Storage Size: " + storageSize + '\n';

**return** computerString;

}

};

*// Function declarations*

computer getComputer();

bool inRange(int input, int startRange, int endRange);

int menu();

void printComputers(vector<computer> &computers);

int main(){

vector<computer> computers;

int computerIndex = 0;

*// Create first computer*

computer newComputer = getComputer();

computers.push\_back(newComputer);

*// Output computer*

cout << "You've created " << computerIndex + 1 << " computers." << endl;

cout << "Here is the computer you just created:" << endl;

cout << computers[computerIndex].toString() << endl;

bool running = true;

**while**(running){

*// Get menu input for main menu*

int menuChoice = menu();

**switch**(menuChoice){

**case** 1:

*// Make new computer, add to vector*

newComputer = getComputer();

computers.push\_back(newComputer);

computerIndex++;

*// Output computer they just made and total num of computers*

cout << endl << "You've created " << computerIndex + 1 << " computers." << endl;

cout << "Here is the computer you just created:" << endl;

cout << computers[computerIndex].toString() << endl;

**break**;

**case** 2:

*// Output computers made so far*

printComputers(computers);

**break**;

**case** 3:

*// Output computers and exit program*

printComputers(computers);

cout << "Goodbye!" << endl;

running = false;

}

}

**return** 0;

}

*// Gets input for main menu*

int menu(){

cout << "Please choose from the following options:" << endl

<< "1. Create a computer" << endl

<< "2. View current computers" << endl

<< "3. Exit Program";

computer newComputer;

int menuChoice;

bool valid = false;

**while**(!valid){

**try**{

menuChoice = getInput<int>("", "Try again. Enter an int between 1 and 3.");

**if**(!inRange(menuChoice, 1, 3)) **throw**(invalid\_argument("Enter an int between 1 and 3."));

valid = true;

}**catch**(**const** exception& e){

cerr << endl << e.what() << endl;

}

}

**return** menuChoice;

}

computer getComputer(){

computer newComputer;

bool valid;

*// Getting manufacturer*

string manufacturer;

manufacturer = getInput<string>("Please enter the manufacturer of your computer: ", "Try again. Enter the manufacturer of your computer.");

newComputer.setManufacturer(manufacturer);

*// Getting form factor with menu*

cout << "Please choose an option for the form factor of your computer:" << endl

<< "1. Desktop" << endl

<< "2. Laptop" << endl;

int formFactorChoice;

valid = false;

**while**(!valid){

**try**{ *// Getting int between 1 and 2*

formFactorChoice = getInput<int>("", "Try again. Enter an int between 1 and 2.");

**if**(!inRange(formFactorChoice, 1, 2)) **throw**(invalid\_argument("Please enter an int between 1 and 2."));

valid = true;

}**catch**(**const** exception& e){

cerr << endl << e.what() << endl;

}

}

string formFactor;

**if**(formFactorChoice == 1){formFactor = "Desktop";}

**else**{formFactor = "Laptop";}

*// Setting form factor*

newComputer.setFormFactor(formFactor);

*// Getting serial number greater than 0*

int serialNum;

valid = false;

**while**(!valid){

**try**{

serialNum = getInput<int>("Please enter the serial number of your computer: ", "Please enter a valid integer.");

newComputer.setSerialNum(serialNum);

valid = true;

}**catch**(**const** exception& e){

cerr << endl << e.what() << endl;

}

}

*// Getting processor*

*// No input validation because there's no invalid processor*

string processor;

processor = getInput<string>("Please enter the processor of your computer: ", "Try again. Enter the processor of your computer.");

newComputer.setProcessor(processor);

int ramChoice;

cout << "Please choose from the following options for your ram size:" << endl

<< "1. 4GB" << endl

<< "2. 6GB" << endl

<< "3. 8GB" << endl

<< "4. 16GB" << endl

<< "5. 32GB" << endl

<< "6. 64GB" << endl;

*// Getting valid menu input*

valid = false;

**while**(!valid){

**try**{

ramChoice = getInput<int>("", "Try again. Enter an integer between 1 and 6.");

**if**(!inRange(ramChoice, 1, 6)) **throw**(invalid\_argument("Please enter an integer between 1 and 6."));

valid = true;

}**catch**(**const** exception& e){

cerr << endl << e.what() << endl;

}

}

int ram;

*// Switch for ram menu*

**switch**(ramChoice){

**case** 1:

ram = 4;

**break**;

**case** 2:

ram = 6;

**break**;

**case** 3:

ram = 8;

**break**;

**case** 4:

ram = 16;

**break**;

**case** 5:

ram = 32;

**break**;

**case** 6:

ram = 64;

**break**;

}

newComputer.setRam(ram);

*// Getting storage type from 3 options*

int storageTypeChoice;

cout << "Please choose from the following options for your storage type:" << endl

<< "1. SSD (Solid State Drive)" << endl

<< "2. HDD (Hard Disk Drive)" << endl

<< "3. UFS (Universal Flash Storage)" << endl;

valid = false;

**while**(!valid){

**try**{

*// Get menu input*

cin >> storageTypeChoice;

**if**(cin.fail() || storageTypeChoice < 1 || storageTypeChoice > 3){

cin.clear();

cin.ignore(10000, '\n');

**throw**(invalid\_argument("Please enter an integer between 1 and 3."));

}

valid = true;

}**catch**(**const** exception& e){

cerr << endl << e.what() << endl;

}

}

*// Setting storage type with static\_cast to enum*

storageTypes storageType = **static\_cast**<storageTypes>(storageTypeChoice);

newComputer.setStorageType(storageType);

*// Getting storage size from 5 options*

int storageSizeChoice;

cout << "Please choose from the following options for your storage size:" << endl

<< "1. 128GB" << endl

<< "2. 256GB" << endl

<< "3. 512GB" << endl

<< "4. 1TB" << endl

<< "5. 2TB" << endl;

valid = false;

**while**(!valid){

**try**{

*// Getting input*

storageSizeChoice = getInput<int>("", "Please enter an int between 1 and 5.");

**if**(storageSizeChoice < 1 || storageSizeChoice > 5) **throw**(runtime\_error("Invalid choice. Please enter an int between 1 and 5."));

valid = true;

}**catch**(**const** exception& e){

cerr << endl << e.what() << endl;

}

}

*// Switch for menu options*

string storageSize;

**switch**(storageSizeChoice){

**case** 1:

storageSize = "128GB";

**break**;

**case** 2:

storageSize = "256GB";

**break**;

**case** 3:

storageSize = "512GB";

**break**;

**case** 4:

storageSize = "1TB";

**break**;

**case** 5:

storageSize = "2TB";

**break**;

}

newComputer.setStorageSize(storageSize);

**return** newComputer;

}

*// Checks if an input is within a range given by startRange and endRange*

bool inRange(int input, int startRange, int endRange){

**if**(input < startRange || input > endRange){**return** false;}

**else**{**return** true;}

}

*// Prints roster of computers*

void printComputers(vector<computer> &computers){

cout << endl << "Here are all of the computers you've created:" << endl << endl;

**for**(int i = 0; i < computers.size(); i++){

cout << computers[i].toString() << endl;

}

}

# 5 Three Use Case Screen Shots

















