## CSCI 140 PA 2 Submission

## Due Date: 09/02/2021 Late (date and time):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Name(s): Nero Li

Exercise 1 -- need to submit source code and I/O  
 -- check if completely done ✔️ ; otherwise, discuss issues below  
Source code below:

/\* Program: PA\_2\_exercise\_1

Author: Nero Li

Class: CSCI 220

Date: 09/02/2021

Description:

Write a C++ class that is derived from the

Progression class to produce a progression

where each value is the absolute value of

the difference between the previous two

values. You should include a default

constructor that starts with 2 and 200 as

the first two values and a parametric

constructor that starts with a specified

pair of numbers as the first two values.

I certify that the code below is my own work.

Exception(s): N/A

\*/

#include <iostream>

using namespace std;

class Progression

{

protected: // member data

long first; // first value of the progression

long cur; // current value of the progression

virtual long firstValue() // reset and return first value

{

cur = first;

return cur;

}

virtual long nextValue() // advance and return next value

{

return ++cur;

}

public:

Progression(long f = 0) // constructor given first value

{

cur = first = f;

}

void printProgression(int n); // print the first n values

virtual ~Progression() { } // virtual destructor

// virtual void print() = 0; // an example of a pure virtual function

};

void Progression::printProgression(int n)

{

std::cout << firstValue(); // print the first value

for (int i = 2; i <= n; i++)

{

std::cout << ' ' << nextValue(); // print values 2 through n

}

std::cout << '\n'; // print end of line

}

class AbsoluteProgression : public Progression

{

protected:

long second;

long prev;

virtual long firstValue()

{

cur = first;

prev = second + first;

return cur;

}

virtual long nextValue()

{

long temp = prev;

prev = cur;

cur -= temp;

if (cur < 0)

cur = -cur;

return cur;

}

public:

AbsoluteProgression(long f = 2, long s = 200)

: Progression(f), second(s), prev(second + first) {}

};

/\*\* Test program \*/

int main()

{

Progression\* prog;

// test AbsoluteProgression

cout << "Absolute progression with default start values:\n";

prog = new AbsoluteProgression();

prog->printProgression(20);

cout << "Absolute progression with start values 4 and 6:\n";

prog = new AbsoluteProgression(4, 6);

prog->printProgression(20);

cout << "Absolute progression with start values 50 and 48:\n";

prog = new AbsoluteProgression(50, 48);

prog->printProgression(20);

cout << "Modified by: Nero Li\n";

return 0;

}

Input/output below:

Absolute progression with default start values:

2 200 198 2 196 194 2 192 190 2 188 186 2 184 182 2 180 178 2 176

Absolute progression with start values 4 and 6:

4 6 2 4 2 2 0 2 2 0 2 2 0 2 2 0 2 2 0 2

Absolute progression with start values 50 and 48:

50 48 2 46 44 2 42 40 2 38 36 2 34 32 2 30 28 2 26 24

Modified by: Nero Li

Exercise 2 -- need to submit source code and I/O  
 -- check if completely done ✔️ ; otherwise, discuss issues below  
Source code below:

/\* Program: PA\_2\_exercise\_2

Author: Nero Li

Class: CSCI 220

Date: 09/02/2021

Description:

Game Information: Provide two classes, GameEntry

and GameScore, to maintain a list of top scores

I certify that the code below is my own work.

Exception(s): Progression

\*/

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;

class GameEntry

{

private:

string name;

int score;

string date;

public:

GameEntry(string name, int score, string date)

: name(name), score(score), date(date)

{

if (score < 0)

{

score = 0;

}

}

void setName(string str)

{

name = str;

}

void setScore(int n)

{

score = n;

if (score < 0)

{

score = 0;

}

}

void setDate(string str)

{

date = str;

}

string getName()

{

return name;

}

int getScore()

{

return score;

}

string getDate()

{

return date;

}

bool operator<(GameEntry \*other)

{

return score < other->score;

}

~GameEntry();

};

class GameScore

{

private:

string name;

GameEntry \*gamer[10];

int size;

void sort()

{

for (int i = 0; i < size - 1; ++i)

{

for (int j = 0; j < size - 1 - i; ++j)

{

if (gamer[j] < gamer[j+1])

{

GameEntry \*temp{gamer[j+1]};

gamer[j+1] = gamer[j];

gamer[j] = temp;

}

}

}

}

public:

GameScore(string name)

: name(name), size(0) {}

void setName(string str)

{

name = str;

}

string getName()

{

return name;

}

void add(GameEntry \*new\_gamer)

{

gamer[size++] = new\_gamer;

}

void print()

{

sort();

cout << "Name: " << getName() << endl;

for (int i = 0; i < size; ++i)

{

cout << left << setw(3) << i + 1;

cout << left << setw(9) << gamer[i]->getName();

cout << left << setw(6) << gamer[i]->getScore();

cout << left << setw(12) << gamer[i]->getDate();

cout << endl;

}

}

};

int main()

{

GameScore g1("Classic Pac-Man");

GameEntry \*e1 = new GameEntry("Jill", 980, "08/05/2021");

GameEntry \*e2 = new GameEntry("Jack", 600, "08/18/2021");

GameEntry \*e3 = new GameEntry("Rob", 875, "07/30/2021");

GameEntry \*e4 = new GameEntry("Rob", 900, "08/02/2021");

g1.add(e1);

g1.add(e2);

g1.add(e3);

g1.add(e4);

g1.print();

cout << "Author: Nero Li\n";

return 0;

}

Input/output below:

Name: Classic Pac-Man

1 Jill 980 08/05/2021

2 Rob 900 08/02/2021

3 Rob 875 07/30/2021

4 Jack 600 08/18/2021

Author: Nero Li

Answer for Question 1:

Inheritance creates an is-a relationship, which makes a program:

* Easier to sort variables as a type.
* For a sort of class, no need to repeat typing some same member functions again.
* Better protection for data in class.

Answer for Question 2:

A C++ function will be set to static binding as default. To change it into dynamic, we need to add the keyword “virtual” before the function declaration.

Static binding is also called early binding. It will get prepared during compilation, and it will run based on the type of variable.

Dynamic binding is also called late binding. It will get prepared during runtime, and it will run based on the type of object.

Extra Credit

Source code below:

/\* Program: PA\_2\_exercise\_2

Author: Nero Li

Class: CSCI 220

Date: 09/02/2021

Description:

Game Information: Load scores from a text file.

Provide two classes, GameEntry and GameScore,

to maintain a list of top scores.

I certify that the code below is my own work.

Exception(s): Progression

\*/

#include <iostream>

#include <iomanip>

#include <string>

#include <fstream>

using namespace std;

class GameEntry

{

private:

string name;

int score;

string date;

public:

GameEntry(string name, int score, string date)

: name(name), score(score), date(date)

{

if (score < 0)

{

score = 0;

}

}

void setName(string str)

{

name = str;

}

void setScore(int n)

{

score = n;

if (score < 0)

{

score = 0;

}

}

void setDate(string str)

{

date = str;

}

string getName()

{

return name;

}

int getScore()

{

return score;

}

string getDate()

{

return date;

}

bool operator<(GameEntry \*other)

{

return score < other->score;

}

~GameEntry();

};

class GameScore

{

private:

string name;

GameEntry \*gamer[10];

int size;

void sort()

{

for (int i = 0; i < size - 1; ++i)

{

for (int j = 0; j < size - 1 - i; ++j)

{

if (gamer[j] < gamer[j+1])

{

GameEntry \*temp{gamer[j+1]};

gamer[j+1] = gamer[j];

gamer[j] = temp;

}

}

}

}

public:

GameScore(string name)

: name(name), size(0) {}

void setName(string str)

{

name = str;

}

string getName()

{

return name;

}

void add(GameEntry \*new\_gamer)

{

gamer[size++] = new\_gamer;

}

void print()

{

sort();

cout << "Name: " << getName() << endl;

for (int i = 0; i < size; ++i)

{

cout << left << setw(3) << i + 1;

cout << left << setw(9) << gamer[i]->getName();

cout << left << setw(6) << gamer[i]->getScore();

cout << left << setw(12) << gamer[i]->getDate();

cout << endl;

}

}

};

int main()

{

ifstream file;

string gameName;

string player;

int score;

string date;

file.open("pacman.txt", ios::binary);

getline(file, gameName);

GameScore g1(gameName);

file >> player >> score >> date;

GameEntry \*e1 = new GameEntry(player, score, date);

g1.add(e1);

file >> player >> score >> date;

GameEntry \*e2 = new GameEntry(player, score, date);

g1.add(e2);

file >> player >> score >> date;

GameEntry \*e3 = new GameEntry(player, score, date);

g1.add(e3);

file >> player >> score >> date;

GameEntry \*e4 = new GameEntry(player, score, date);

g1.add(e4);

g1.print();

cout << "Author: Nero Li\n";

return 0;

}

Input/output below:  
  
Name: Classic Pac-Man

1 Jill 980 08/05/2021

2 Rob 900 08/02/2021

3 Rob 875 07/30/2021

4 Jack 600 08/18/2021

Author: Nero Li