Amir Sotoodeh
CS256
5/1/18

Assignment #4

Github: https://github.com/excisionhd/CS256/blob/master/hwk4.cpp

Output (See page 3 for source code)

```
Homework Assignment #4
Chapter 13 Problem 18: A Game of 21.

Welcome to 21
-------------
Computer has rolled.
Your Points: 0
Type y to roll, type n to stay.
y
You rolled 6.
Computer has rolled.
Your Points: 6
Type y to roll, type n to stay.
y
You rolled 2.
Computer has rolled.
Your Points: 8
Type y to roll, type n to stay.
y
You rolled 3.
Computer has rolled.
Your Points: 11
Type y to roll, type n to stay.
n
Your Points: 11
Computer's Points: 13
You lose!

Chapter 14: Problem #14 Parking Ticket Sim

Bob Smith has issued a ticket.
Ticket details:
Make: Chevrolet
Model: Corvette
Color: Black
License Number: 7NAL382
Time Stayed: 121 minutes
Meter Reading: 60 minutes
Officer Name: Bob Smith
```

```
Officer Badge Number: 538493
Fine: $35

Donald Trump has issued a ticket.
Ticket details
--------------
Make: BMW
Model: M3
Color: Black
License Number: 3YNS748
Time Stayed: 186 minutes
Meter Reading: 60 minutes
Officer Name: Donald Trump
Officer Badge Number: 985243
Fine: $45

Chapter 15: Problem #12  Ship, Cruise, Cargo Ship

Ships
-----
Name: Windsong
Year: 1954

Name: Apothecary
Year: 1932
Tonnage: 300

Name: Legendary
Year: 1984
Number of Passengers: 1204

Name: The SHIP
Year: 1893

Press any key to continue . . .
```

```
/*****************************************************
* FILENAME : hwk4.cpp
*
* DESCRIPTION : Programming Assignment #4 dealing with
* classes; concepts such as inheritance, method
* overloading, virtuals, and polymorphisms are
* dealt with.
* AUTHOR :  Amir Sotoodeh
* START DATE : 5/5/18
*
*****************************************************/

#include <stdafx.h>
#include <cstdlib> // For rand and srand
#include <ctime> // For the time function
#include <iostream>
#include <string>
#include <windows.h>

using namespace std;

//die class imported from book
class Die {
	private:
		int sides;
		int value;
	public:
		Die(int numSides){
			unsigned seed = time(0);
			srand(seed);
			sides = numSides;
			roll();
		}
		void roll(){
			const int MIN_VALUE = 1;
			value = (rand() % (sides - MIN_VALUE + 1)) + MIN_VALUE;
			}
		int getSides(){
			return sides;
		}

		int getValue(){
			return value;
		}
};
```

```
class ParkedCar {
	private:
		string make, model, color, licenseNum;
		int minutes;
	public:
		ParkedCar(string m, string mo, string co, string license, int min) {
			make = m;
			model = mo;
			color = co;
			licenseNum = license;
			minutes = min;
		}
		string getMake() {
			return make;
		}
		string getModel() {
			return model;
		}
		string getColor() {
			return color;
		}
		string getLicenseNumber() {
			return licenseNum;
		}
		int getMinutes() {
			return minutes;
		}
};

class ParkingMeter {
	private:
		int minutes;
	public:
		ParkingMeter(int min) {
			minutes = min;
		}
		int getMinutes(){
			return minutes;
		}
};

class PoliceOfficer {
private:
	string name;
```

```
        int badgeNumber;
public:
        PoliceOfficer(string n, int badge) {
                name = n;
                badgeNumber = badge;
        }

        //return true if car stayed longer than meter
        bool examine(ParkedCar p, ParkingMeter m) {
                return (p.getMinutes() > m.getMinutes());
        }

        string getName() {
                return name;
        }

        int getBadgeNumber() {
                return badgeNumber;
        }

};

class ParkingTicket {
private:
        string make, model, color, licenseNum, name;
        int fine, badgeNumber, carMins, meterMins;
public:
        ParkingTicket(ParkedCar car, ParkingMeter meter, PoliceOfficer po) {
                make = car.getMake();
                model = car.getModel();
                color = car.getColor();
                licenseNum = car.getLicenseNumber();
                carMins = car.getMinutes();
                meterMins = meter.getMinutes();
                name = po.getName();
                badgeNumber = po.getBadgeNumber();

                if ((carMins - meterMins) <= 60) {
                        fine = 25;
                }
                else {
                        int additionalHours = (int)ceil((double)((carMins - 60) / 60));
                        fine = (additionalHours * 10) + 25;
                }
        }
```

```
        void print() {
                cout << "Make: " << make << endl;
                cout << "Model: " << model << endl;
                cout << "Color: " << color << endl;
                cout << "License Number: " << licenseNum << endl;
                cout << "Time Stayed: " << carMins << " minutes" << endl;
                cout << "Meter Reading: " << meterMins << " minutes" << endl;
                cout << "Officer Name: " << name << endl;
                cout << "Officer Badge Number: " << badgeNumber << endl;
                cout << "Fine: $" << fine << endl;
                cout << endl;
        }
};

class Ship {
        protected:
                string name;
                string year;
        public:
                Ship(string n, string y) {
                        name = n;
                        year = y;
                }
                Ship() {
                        name = " ";
                        year = "0";
                }
                virtual void print() {
                        cout << "Name: " << name << endl;
                        cout << "Year: " << year << endl;
                }
                void setName(string n) {
                        name = n;
                }
                void setYear(string y) {
                        year = y;
                }
                string getName() {
                        return name;
                }
                string getYear() {
                        return year;
                }
};
```

```
class CruiseShip : public Ship {
	private:
		int numPassengers;
	public:
		CruiseShip(int n) {
			numPassengers = n;
		}
		CruiseShip(string n, string y, int p) {
			name = n;
			year = y;
			numPassengers = p;
		}
		virtual void print() override{
			cout << "Name: " << getName() << endl;
			cout << "Year: " << getYear() << endl;
			cout << "Number of Passengers: "<< numPassengers << endl;
		}
		void setPassengers(int p) {
			numPassengers = p;
		}
		int getPassengers() {
			return numPassengers;
		}
};

class CargoShip : public Ship {
	private:
		int tonnage;
	public:
		CargoShip(int n) {
			tonnage = n;
		}
		CargoShip(string n, string y, int t) {
			name = n;
			year = y;
			tonnage = t;
		}
		virtual void print() override{
		cout << "Name: " << getName() << endl;
		cout << "Year: " << getYear() << endl;
		cout << "Tonnage: " << tonnage << endl;
		}
		void setTonnage(int t) {
		tonnage = t;
```

```
		}
		int getTonnage() {
		return tonnage;
		}
};


int main() {
	HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
	SetConsoleTextAttribute(hConsole, 13);

	cout << "Homework Assignment #4" << endl;

	cout << "Chapter 13 Problem 18: A Game of 21." << endl << endl;
	bool play = true;
	char input;
	Die d1(6); //Two six sided die
	Die d2(6);
	int playerTotal = 0;
	int botTotal = 0;

	cout << "Welcome to 21" << endl;
	cout << "-------------" << endl;
	while (play) {
		d1.roll();
		botTotal += d1.getValue();
		cout << "Computer has rolled." << endl;
		if (botTotal > 21) {
			cout << "Your Points: " << playerTotal << endl;
			cout << "Computer's Points: " << botTotal << endl;
			cout << "Computer busted!  You win." << endl;
			exit(0);  //exit to avoid break print statement.
		}
		cout << "Your Points: " << playerTotal << endl;
		cout << "Type y to roll, type n to stay." << endl;
		cin >> input;

		if (input == 'y') {
			d2.roll();
			playerTotal += d2.getValue();
			if (playerTotal > 21) {
				cout << "You rolled " << d2.getValue() << "." << endl;
				cout << "Your Points: " << playerTotal << endl;
				cout << "Computer's Points: " << botTotal << endl;
				cout << "You lose." << endl;
```

```
                        exit(0); //exit to avoid print statement
                }
                else {
                        cout << "You rolled " << d2.getValue() << "." << endl;
                }
        }
        else if(input == 'n') {
                play = false;
        }
}
        if (playerTotal > botTotal) {
                cout << "Your Points: " << playerTotal << endl;
                cout << "Computer's Points: " << botTotal << endl;
                cout << "You win!" << endl;
                play = false;
        }
        else {
                cout << "Your Points: " << playerTotal << endl;
                cout << "Computer's Points: " << botTotal << endl;
                cout << "You lose!" << endl;
                play = false;
        }

cout << endl << "Chapter 14: Problem #14 Parking Ticket Sim" << endl << endl;
ParkedCar c("Chevrolet", "Corvette", "Black", "7NAL382", 121);
ParkingMeter m(60);
PoliceOfficer p("Bob Smith", 538493);
if (p.examine(c, m) == true) {
        cout << p.getName() << " has issued a ticket." << endl;
        cout << "Ticket details: " << endl;
        ParkingTicket t(c, m, p);
        t.print();
}

ParkedCar c2("BMW", "M3", "Black", "3YNS748", 186);
ParkingMeter m2(60);
PoliceOfficer p2("Donald Trump", 985243);
if (p2.examine(c2, m2) == true) {
        cout << p2.getName() << " has issued a ticket." << endl;
        cout << "Ticket details" << endl;
        cout << "--------------" << endl;
        ParkingTicket t2(c2, m2, p2);
        t2.print();
}
```

```
	cout << "Chapter 15: Problem #12  Ship, Cruise, Cargo Ship" << endl << endl;
	const int NUM_SHIPS = 4;

	Ship *ships[NUM_SHIPS] = {
		new Ship("Windsong","1954"),
		new CargoShip("Apothecary", "1932", 300),
		new CruiseShip("Legendary", "1984", 1204),
		new Ship("The SHIP", "1893")
		};

	cout << "Ships" << endl;
	cout << "-----" << endl;
	for (int i = 0; i < NUM_SHIPS; i++) {
		ships[i]->print();
		cout << endl;
	}

	return 0;

}
```