TEST.CPP

#include "Zoo.h"

#include "Giraffe.h"

#include "Tiger.h"

#include "Animal.h"

//liron hamo 207973603

#include "Bat.h"

#include <iostream>

#include <string>

using namespace std;

int main() {

cout << "liron hamo 207973603\n";

Zoo safari("safari");

Animal\* animal = NULL;

int size, type, choice, idr,animalnum\_7;

string name, nickname, department, foodbat;

float age, FoodPerDay,maxSpeed,height;

do {

cout << "to add animal press 1\nto remove animal press 2\nto get data about animal press 3\nto feed animals press 4\nto sleeping the animals press 5\nto print zoo press 6\nto print how much from type press 7\nto exit press 8\n";

cin >> choice;

switch (choice) {

case 1:

cout << "how many animals you want?\n";

cin >> size;

for (size\_t i{ 0 };i < size;i++) {

cout << "for tiger: 1\n for giraffe: 2\n for bat:3\n ";

cin >> type;

if (type > 0 && type < 4) {//check if the value is corect

cout << "enter the name,nickname, age,department of the animal and food per day\n";

cin >> name >> nickname >> age >> department >> FoodPerDay;

if (type == 1) {//build a obj of tiger

cout << "enter the max speed of the tiger\n";

cin >> maxSpeed;

auto tiger = new Tiger(maxSpeed, name, nickname, age, department, FoodPerDay);

safari.addAnimal(tiger);

}

if (type == 2) {//build giraffe obj

cout << "enter the hegiht of the giraffe\n";

cin >> height;

auto giraffe = new Giraffe(height, name, nickname, age, department, FoodPerDay);

safari.addAnimal(giraffe);

}

if (type == 3) {//build bat obj

cout << "enter the type of food of the bat\n";

cin >> foodbat;

auto bat = new Bat(foodbat, name, nickname, age, department, FoodPerDay);

safari.addAnimal(bat);

}

}

else cout<<"your choice must be from 1-3\n";//the coice is illegal

}

break;

case 2:

cout << "what the id you want to remove?\n";//remove animal from the zoo

cin >> idr;

safari.removeAnimal(idr);

break;

case 3:

cout << "what the id you want to get?\n";//get data

cin >> idr;

animal=safari.getAnimal(idr);

break;

case 4:

safari.feedAllAnimals();

break;

case 5:

safari.sleepAnimals();

break;

case 6:

safari.PrintZoo();

break;

case 7:

cout << "enter the num of the animal you want to see : 1==tiger,2==giraffe,3==bat\n";

cin >> animalnum\_7;

safari.print\_type(animalnum\_7);

break;

case 8:

cout << "goodbye\n";

break;

default:

cout << "choice must be from 0 to 8\n\n";

}

} while (choice !=8);

system("pause");

return 0;

}

BAT.cpp

//liron hamo 207973603

#include "Bat.h"

#include <array>

#include <iostream>

#include <string>

using namespace std;

Bat::Bat(string food, string name, string nickname, float age, const string department, float FoodPerDay):

Animal { name, nickname,age,department,FoodPerDay }//constructor by parmeters

{

this->food = food;

}

Bat::Bat(Bat& other) : Animal(other) {//copy constructor

this->food = other.food;

}

void Bat::eat() {//what bat eat

cout << "Bat" << id << " eats every day " << FoodPerDay << "Kg of" << food<< endl;

}

void Bat::sleep() {

cout << " Bat" << id << " sleeps only at day\n";

}

void Bat::print() {

Animal::print();//print all data from animal

cout << "it food:" << food<< endl;//print the spcial data

}

Bat::~Bat()

{

}

ZOO.h

//liron hamo 207973603

#pragma once

#include "Animal.h"

#include <vector>

#include <string>

class Zoo

{

public:

Zoo(string name);//constructor

const string getName() { return name; };

Animal\*\* getAnimalsArr() { return animalArr; };

void addAnimal(Animal\* animal);//add animal to the zoo

Animal\* getAnimal(int id);

bool removeAnimal(int id);

void feedAllAnimals() const;

void sleepAnimals() const;

void PrintZoo() const;

void print\_type(int t); //t==1: tiger, t==2: giraffe, t==3: bat

~Zoo();//destructor

private:

const std::string name;//name of the zoo

Animal\*\* animalArr;

int sizeArr;//how much animals at the zoo

};

BAT.h

//liron hamo 207973603

#pragma once

#include <string>

#include "Animal.h"

class Bat: public Animal

{

public:

Bat(string food, string name, string nickname, float age, const string department, float FoodPerDay);

Bat(Bat& other);//copy constructor

const string getfood() { return food; };

void eat();//print eat

void sleep();//print sleep

void print();//print data

~Bat();

private:

std::string food; //special data

};

Zoo.cpp

//liron hamo 207973603

#include "Zoo.h"

#include "Tiger.h"

#include "Bat.h"

#include "Giraffe.h"

#include "Animal.h"

#include <array>

#include <string>

#include <iostream>

using namespace std;

Zoo::Zoo(string name) : name{ name }

{

animalArr = NULL;

sizeArr = 0;

}

void Zoo::addAnimal(Animal\* animal) {

Animal\*\* tmp = NULL;

tmp = new Animal\*[sizeArr + 1];

for (size\_t i{ 0 };i < sizeArr;i++) {

tmp[i] = animalArr[i];

}

tmp[sizeArr] = animal;

sizeArr++;

delete[] animalArr;

animalArr = tmp;

}

Animal\* Zoo::getAnimal(int id) {

for (size\_t i{ 0 };i < sizeArr;i++) {

if (animalArr[i]->getId() == id)

return animalArr[i];

}

return 0;

}

bool Zoo::removeAnimal(int id) {

for (size\_t i{ 0 };i < sizeArr;i++) {

if (animalArr[i]->getId() == id) {

animalArr[i] = animalArr[sizeArr - 1];

sizeArr--;

cout << "the animal removed\n" << endl;

return true;

}

}

return false;

}

void Zoo::feedAllAnimals() const {//print what all animals eat

for (size\_t i{ 0 };i < sizeArr;i++) {

animalArr[i]->eat();

}

}

void Zoo::sleepAnimals() const {//when the animals sleep

for (size\_t i{ 0 };i < sizeArr;i++) {

animalArr[i]->sleep();

}

}

void Zoo::PrintZoo() const {//print all animals at the zoo

cout << name << endl << "size of animals:" << sizeArr << endl;

for (size\_t i{ 0 };i < sizeArr;i++) {

if (auto a = dynamic\_cast<Tiger\*>(animalArr[i])) {

a->print();

}

if (auto a = dynamic\_cast<Giraffe\*>(animalArr[i])) {

a->print();

}

if (auto a = dynamic\_cast<Bat\*>(animalArr[i])) {

a->print();

}

}

}

void Zoo::print\_type(int t) {//t==1: tiger, t==2: giraffe, t==3: bat

int tiger = 0, giraffe = 0, bat = 0;

if (t == 1) {

for (size\_t i{ 0 };i < sizeArr;i++) {

Animal\* tmp = animalArr[i];

if (dynamic\_cast<Tiger\*>(tmp)) {

tmp->print();

cout << endl;

}

}

}

else if (t == 2) {

for (size\_t i{ 0 };i < sizeArr;i++) {

Animal\* tmp = animalArr[i];

if (dynamic\_cast<Giraffe\*>(tmp)) {

giraffe++;

tmp->print();

cout << endl;

}

}

}

else if (t == 3) {

for (size\_t i{ 0 };i < sizeArr;i++) {

Animal\* tmp = animalArr[i];

if (dynamic\_cast<Bat\*>(tmp)) {

bat++;

tmp->print();

cout << endl;

}

}

}

}

Zoo::~Zoo()

{

for (int i = 0; i < sizeArr;i++) {

delete[] this->animalArr[i];

}

sizeArr = 0;

}

Tiger.cpp

//liron hamo 207973603

#include "Tiger.h"

#include "Animal.h"

#include <iostream>

#include <string>

using namespace std;

Tiger::Tiger(const float maxSpeed, string name, string nickname, float age, const string department, float FoodPerDay) :

Animal(name, nickname, age, department, FoodPerDay), maxSpeed{ maxSpeed }// prameters constuctor

{

}

Tiger::Tiger(Tiger& other) : Animal(other),maxSpeed{other.maxSpeed}//copy constructor

{

}

void Tiger::eat(){

cout << "Tiger" << id << " eats every day " << FoodPerDay << "Kg meat\n" << endl;

}

void Tiger::sleep() {

cout << "Tiger" << id << "sleeps only at night\n";

}

void Tiger::print(){//print data of the tiger

this->Animal::print();

cout << " it max speed:" << maxSpeed<< endl;//print the special data

}

Tiger::~Tiger()

{

}

GIRAFFE.cpp

//liron hamo 207973603

#include "Giraffe.h"

#include <iostream>

#include <string>

using namespace std;

Giraffe::Giraffe(float height, string name, string nickname, float age, const string department, float FoodPerDay):

Animal (name,nickname,age, department,FoodPerDay)//parameters constructor

{

this->height = height;

}

Giraffe::Giraffe(Giraffe& other) : Animal(other) {//copy constructor

this->height = other.height;

}

void Giraffe::eat() {//what giraffe eat and haw much

cout << "Giraffe" << id << " eats every day " << FoodPerDay << " Kg leaves\n" << endl;

}

void Giraffe::sleep() {

cout << "Giraffe" << id << " sleeps both day and night\n";

}

void Giraffe::print() {//print data

Animal::print();

cout << "it height:" << height<< endl;

}

Giraffe::~Giraffe()

{

}

Animal.cpp

//liron hamo 207973603

#include "Animal.h"

#include <iostream>

#include <array>

#include <string>

using namespace std;

int Animal::count{ 0 };

Animal::Animal(string name, string nickname, float age, const string department, float FoodPerDay): id{++count}, department{ department }//parmeters constructor

{

this->name = name;

this->nickname = nickname;

this->age = age;

this->FoodPerDay = FoodPerDay;

}

Animal::Animal(Animal &other) : id{ other.id } , department{ other.department }//copry constuctor

{

this->name = other.name;

this->nickname = other.nickname;

this->age = other.age;

this->FoodPerDay = other.FoodPerDay;

}

void Animal::print() {//print data of the animal

cout << "Animal department:" << this->department << endl <<

"Animal name:" << this->name << endl <<

"Animal nickname:" << this->nickname << endl <<

"Animal id:" << this->id << endl <<

"Animal age:" << this->age << endl;

}

Animal::~Animal()

{

}

Tiger.h

//liron hamo 207973603

#pragma once

#include "Animal.h"

class Tiger :public Animal

{

public:

Tiger(float maxSpeed,string name, string nickname, float age, const string department, float FoodPerDay);

Tiger(Tiger& other);//copy constructor

const float getMaxspeed() { return maxSpeed; };

void eat();

void sleep();

void print();

~Tiger();

private:

const float maxSpeed;//special data

};

Giraffe.h

//liron hamo 207973603

#pragma once

#include "Animal.h"

class Giraffe :public Animal

{

public:

Giraffe(float height, string name, string nickname, float age, const string department, float FoodPerDay);

Giraffe(Giraffe& other);//copy constructor

float getHeight() { return height; };

void eat();

void sleep();

void print();

~Giraffe();

private:

float height;//special data

};

Animal.h

//liron hamo 207973603

#pragma once

#include <string>

using namespace std;

class Animal

{

public:

Animal(string name, string nickname, float age, const string department, float FoodPerDay);

Animal(Animal &other);// copy

string getName() { return name; };

string getNickName() { return nickname; };

float getAge() { return age; };//age of the animal

const string getDepartment(){ return department; };//class of the animal

float getFoodPerDay() { return FoodPerDay; };

const int getId() { return id; };

virtual void eat() = 0;//the fun is in the heirs

virtual void sleep() = 0;//the fun is in the heirs

void print();//הדפסת פרטים ללא כמות אוכל יומית

virtual ~Animal();

static int count;//static const to change the id

protected:

std::string name;

std::string nickname;

float age;

const std::string department;//מחלקה

float FoodPerDay;

const int id;

};