# Національний технічний університет України

# «Київський політехнічний інститут»

# Факультет інформатики та обчислювальної техніки

# Кафедра обчислювальної техніки

# Лабораторна робота №6

### З програмування

виконав студент першого курсу

група ІО-01

Шийка Владислав Володимирович

Тема: Робота з абстрактними класами та інтерфейсами в мові програмування Java.

Мета: Здобуття навичок у створенні та використанні абстрактних класів та інтерфейсів в мові програмування Java.

1. Визначення варіанту та завдання

Номер залікової книжки – 129

С12=129 mod 2 = 1

Отже завдання має такий вигляд:

Абстрактний клас

1. Використовуючи варіант попередньої роботи створити батьківський клас (п.2) не менше ніж для 3 класів, які описують різновиди батьківського, та складаються не менше ніж з 7 полів (принаймні 2 поля повинні бути агрегованими об’єктами) та методів доступу до цих полів. В класі необхідно описати, використовуючи стандартні засоби мови програмування, не менше 2 методів, за допомогою яких можна було б сортувати масив об’єктів батьківського класу за одним із полів. Для перевірки необхідно створити клас, що складається з виконавчого методу. Всі початкові дані задаються у виконавчому методі.
2. Лістинг додактового класу для створення агрегованого об’єкту Date :

public class Date { //Additional class to use date fields

private int day;

private int month;

private int year;

Date(int day,int month,int year){

//full constructor

setDay(day);

setMonth(month);

setYear(year);

}

Date(){

//empty constructor

this(0,0,0);

}

//Access methods:

public void setDate(int day,int month,int year){

setDay(day);

setMonth(month);

setYear(year);

}

public void setDay(int day) {

this.day = day;

}

public void setMonth(int month) {

this.month = month;

}

public void setYear(int year) {

this.year = year;

}

//Getting methods:

public String getDate(){

return day+"/"+month+"/"+year;

}

public int getDay() {

return day;

}

public int getMonth() {

return month;

}

public int getYear() {

return year;

}

}

1. Лістинг додактового класу для створення агрегованого об’єкту Name :

public class Name {//additional class

private String namestr;

Name(){//empty constructor

namestr="NO\_NAME";

}

Name(String name){//full constructor

setName(name);

}

//Access methods:

//Setting:

public final void setName(String name){

namestr=name;

}

//Getting:

public final String getName(){

return namestr;

}

}

1. Лістинг додактового класу для створення агрегованого об’єкту Model :

public class Model {//additional class

private String modelstr;

Model(){//empty constructor

modelstr="NO\_MODEL";

}

Model(String modelx){//full constructor

setModel(modelx);

}

//Access methods:

//Setting:

public final void setModel(String modelx){

modelstr=modelx;

}

//Getting:

public final String getModel(){

return modelstr;

}

}

1. Лістинг додактового класу для створення агрегованого об’єкту Status :

public class Status {

String statusstr;

Status(String stat){

setStatus(stat);

}

Status(){

this("NO\_STATUS");

}

public void setStatus(String stat){

this.statusstr=stat;

}

public String getStatus(){

return statusstr;

}

}

1. Лістинг батьківського класу:

abstract class AbsPlane {// Parent class

//Common fields:

protected int speed;

protected int distance;

protected int prodyear;

protected int places;

protected int height;

protected Name name = new Name();

protected Status status = new Status();

AbsPlane(int s,int d,int py,int pl,int h,String n,String st){

//full constructor

setSpeed(s);

setDistance(d);

setProdyear(py);

setPlaces(pl);

setHeight(h);

name.setName(n);

status.setStatus(st);

}

AbsPlane(){//empty constructor

this(0,0,0,0,0,"-/-","-/-");

}

AbsPlane(int s,int d,int y,int p){// with setting 4 field

this(s,d,y,p,0,"-/-","-/-");

}

//Access methods:

public void setSpeed(int speed) {

this.speed = speed;

}

public void setDistance(int distance) {

this.distance = distance;

}

public void setProdyear(int prodyear) {

this.prodyear = prodyear;

}

public void setPlaces(int places) {

this.places = places;

}

public void setHeight(int height) {

this.height = height;

}

public void setName(String namestr){

name.setName(namestr);

}

public void setStatus(String statstr){

status.setStatus(statstr);

}

//Getting methods:

public int getSpeed() {

return speed;

}

public int getDistance() {

return distance;

}

public int getProdyear() {

return prodyear;

}

public int getPlaces() {

return places;

}

public int getHeight() {

return height;

}

public String getName(){

return name.getName();

}

public String getStatus(){

return status.getStatus();

}

abstract public void sortByName(AbsPlane[] array);

abstract public void sortBySpeed(AbsPlane[] array);

public static void staticSortByName(AbsPlane [] array){

Arrays.sort(array,new CompareSpeed());

}

public static void staticSortBySpeed(AbsPlane [] array){

Arrays.sort(array,new CompareSpeed());

}

}

1. Лістинг першого дочірнього класу:

public class Techinfo extends AbsPlane {//Child class

//Special fields:

private Model model = new Model();

private Date dateofproduse = new Date();

Techinfo(){//empty constructor

this(0,0,0,0,0,"NO\_NAME","NO\_STATUS","NO\_MODEL",0,0,0);

}

Techinfo(int s ,int d,int py,int pl,int h,String n,String st,

String m,int day,int month,int year){

//full constructor

super(s ,d, py, pl, h, n, st);

setModel(m);

setDate(day, month, year);

}

//Access methods:

public void setModel(String modelstr){

model.setModel(modelstr);

}

public void setDate(int day,int month,int year){

dateofproduse.setDate(day,month,year);

}

//Getting methods:

public String getModel(){

return model.getModel();

}

public String getDate(){

return dateofproduse.getDate();

}

public void sortBySpeed(AbsPlane[] array){

Arrays.sort(array,new CompareSpeed());

}

public void sortByName(AbsPlane[] array){

Arrays.sort(array,new CompareName());

}

}

1. Лістинг другого дочірнього класу:

public class Airline extends AbsPlane {//Child class

//Special fields:

private int ticketprice;

private Date dateofflight = new Date();

Airline(int s ,int d,int py,int pl,int h,String n,String st,

int tp,int day,int month,int year){

//full constructor

super(s,d,py,pl,h,n,st);

setTicketPrice(tp);

setDateOfFlight(day,month,year);

}

Airline(){//empty constructor

this(0,0,0,0,0,"-/-","-/-",0,0,0,0);

}

//Access methods:

public void setTicketPrice(int ticketprice){

this.ticketprice=ticketprice;

}

public void setDateOfFlight(int day,int month,int year){

dateofflight.setDate(day, month, year);

}

//Getting methods:

public String getTicketPrice(){

return ticketprice+"$";

}

public String getDateOfFlight(){

return dateofflight.getDay()+"/"+dateofflight.getMonth()+"/"+dateofflight.getYear();

}

public void sortBySpeed(AbsPlane[] array){

Arrays.sort(array,new CompareSpeed());

}

public void sortByName(AbsPlane[] array){

Arrays.sort(array,new CompareName());

}

}

1. Лістинг третьго дочірнього класу:

public class Additional extends AbsPlane {

//Special fields:

private String country = new String();

private int planeprice;

Additional(int s,int d,int py,int pl,int h,

String n,String st,String c,int pp){

//full constructor

super(s,d,py,pl,h,n,st);

setPlanePrice(pp);

setCountryBuilder(c);

}

Additional(){//empty constructor

this(0,0,0,0,0,"NO\_NAME","NO\_STATUS","NO\_COUNTRY",0);

}

//Access methods:

public void setPlanePrice(int planeprice) {

this.planeprice = planeprice;

}

public void setCountryBuilder(String country){

this.country=country;

}

//Getting methods:

public String getPlanePrice() {

return planeprice+"$";

}

public String getCountryBuilder(){

return country;

}

public void sortBySpeed(AbsPlane[] array){

Arrays.sort(array,new CompareSpeed());

}

public void sortByName(AbsPlane[] array){

Arrays.sort(array,new CompareName());

}

}

1. Лістинг компараторів:

//Compare objects by field "Name"

public class CompareName implements Comparator <AbsPlane> {

public int compare(AbsPlane ob1,AbsPlane ob2){

return ob1.getName().compareTo(ob2.getName());

}

}

//Compare objects by field "Speed"

public class CompareSpeed implements Comparator <AbsPlane> {

public int compare(AbsPlane ob1, AbsPlane ob2) {

return ob1.getSpeed()>ob2.getSpeed()?1:-1;

}

}

1. Лістинг виконавчого класу:

public class ExecLab4 {

public static void main(String[] args) {

AbsPlane[] array= new AbsPlane[3];//Creating array of parent class

//Declare examples of children classes with all information

Techinfo tech1 = new Techinfo(650,18000,2008,660,8000,"T-777","NICE","Enhanced",24,12,2008);

Techinfo tech2 = new Techinfo(680,17500,2009,675,8500,"T-787","GOOD","Modern",5,4,2010);

Techinfo tech3 = new Techinfo(500,12000,2001,570,7500,"T-474","BAD","Standard",21,8,1999);

array[0]=tech1;

array[1]=tech2;

array[2]=tech3;

Airline air = new Airline(800,15000,2001,750,8000,"T-777-M","GOOD",400,21,4,2011);

Additional add = new Additional(900,12000,2010,800,7000,"ABC-47","PERFECT","Ukraine",4750000);

//Displaying info about planes:

System.out.println("----------------------------------");

System.out.println("Techinfo Plane #1:");

System.out.println("Speed: "+tech1.getSpeed());

System.out.println("Distance: "+tech1.getDistance());

System.out.println("Year: "+tech1.getProdyear());

System.out.println("Places :"+tech1.getPlaces());

System.out.println("Heigth :"+tech1.getHeight());

System.out.println("Name :"+tech1.getName());

System.out.println("Status :"+tech1.getStatus());

System.out.println("Model :"+tech1.getModel());

System.out.println("----------------------------------");

System.out.println("Techinfo Plane #2:");

System.out.println("Speed: "+air.getSpeed());

System.out.println("Distance: "+air.getDistance());

System.out.println("Year: "+air.getProdyear());

System.out.println("Places :"+air.getPlaces());

System.out.println("Heigth :"+air.getHeight());

System.out.println("Name :"+air.getName());

System.out.println("Status :"+air.getStatus());

System.out.println("Ticket Price :"+air.getTicketPrice());

System.out.println("Date Of Flight :"+air.getDateOfFlight());

System.out.println("----------------------------------");

System.out.println("Techinfo Plane#3:");

System.out.println("Speed: "+add.getSpeed());

System.out.println("Distance: "+add.getDistance());

System.out.println("Year: "+add.getProdyear());

System.out.println("Places :"+add.getPlaces());

System.out.println("Heigth :"+add.getHeight());

System.out.println("Name :"+add.getName());

System.out.println("Status :"+add.getStatus());

System.out.println("Country Builder :"+add.getCountryBuilder());

System.out.println("Plane Price :"+add.getPlanePrice());

System.out.println("----------------------------------");

//Sorting object by field "Speed"

System.out.println("Sorting by speed: ");

tech1.sortByName(array);

for(int i=0;i<3;i++){//Displaying sorted array

System.out.println(array[i].getSpeed());

}

System.out.println("----------------------------------");

//Sorting object by field "Speed"

tech1.sortBySpeed(array);

System.out.println("Sorting by name: ");

for(int i=0;i<3;i++){//Displaying sorted array

System.out.println(array[i].getName());

}

System.out.println("----------------------------------");

array[0]=tech3;

array[1]=air;

array[2]=add;

AbsPlane.staticSortByName(array);

System.out.println("Sorting by name: ");

for(int i=0;i<3;i++){//Displaying sorted array

System.out.println(array[i].getName());

}

System.out.println("----------------------------------");

}

}

**Висновок:** під час виконання цієї лабораторної роботи я здобув навички у створенні та використанні абстрактних класів та інтерфейсів. Здобув навички у створенні та використанні компараторів,у використанні наслідування. Покращив навички у створенні методів доступу,та директив доступу (видимості). Навчився сортувати масив об’єктів за допомогою компараторів, у заданому порядку,за одним з полів.

**Опис:**

В программі використовуються 11 класів.Execlab4 містить тільки виконавчий метод.Name містить поля з інформацією про ім’я літака ,конструктори з та без параметрів,та методами доступу до цих полів.Status містить поля з інформацією про стан літака, конструктори з та без параметрів,та методами доступу до полів.Model містить поля з інформацією про модель літака,конструктори з та без параметрів,методи доступу до цих полів.Date містить поля з інформацією про дату ,конструктори з та без параметрів,та методи доступу до полів.AbsPlane – абстрактний клас,що містить поля з інформацією та агреговані об’єкти класів Name і Status. Містить конструктори з та без параметрів,методи доступу до цих полів. Класи Techinfo, Airline,Additional – дочірні класи класу AbsPlane,що реалізують всі його методи. Класи CompareName i CompareSpeed компаратори ,за допомогою яких відбувається сортування масиву об’єктів класу AbsPlane у заданому порядку за полем Name та за полем Speed відповідно.