****

Latvijas Universitātes Vadības un uzņēmējdarbības mācību centrs

(LU VUMC)

**IEVADS JAVA PROGRAMMĒŠANĀ**

**NODARBĪBA #8**

Mārtiņš Ceske/Vjačeslavs Vasiļevskis

2021

ESF projekts Nr. 8.4.1.0/16/l/001  
‘’Nodarbināto personu profesionālās kompetences pilnveide”



# 1. mājas darbs

# OOP pamati

### **Noteikumi:**

* Izveidojiet klases hierarhiju. Galvenajai hierarhijas klasei jābūt klase Mašīna (Car).
* Klases “Car” mantinieki:
  + KravasAutomašīna (Truck)
  + Autobuss (Buss)
* Izveidojam interfeisu Passanger, kuru klase Buss implementēs. Iterfeisam Passanger ir jābūt metodei passengerSeatCount()
* Kopīgajām īpašībām jābūt klasē “Car”, bet citas īpašības (brīvi izvēlamies) ir jānorāda mantiniekos.
* Katrā klasē pārrakstiet toString() un equals() metodes.
* Demonstrēt klašu darbību izveidojot objektus no tām un izdrukājot tās konsolē

### **Soļi:**

* Izveidojam nepieciešamās klases un interfeisu

public class Car {  
 }  
  
 public class Truck extends Car {  
 }  
  
 public class Buss extends Car implements Passenger {  
 }  
  
 public interface Passenger {  
 }

* Pievienojam konstruktoru katrai klasei un attiecīgi laukus

public class Car {  
  
 private final String color;  
 private final int year;  
 private final String manufacturer;  
  
 public Car(String color, int year, String manufacturer) {  
 this.color = color;  
 this.year = year;  
 this.manufacturer = manufacturer;  
 }  
}  
  
 public class Truck extends Car {  
  
 private long loadCapacity;  
  
 public Truck(String color, int year, String manufacturer, long loadCapacity) {  
 super(color, year, manufacturer);  
 this.loadCapacity = loadCapacity;  
 }  
 }

* Pievienojam interfeisam metodi passengerSeatCount un attiecīgo implementāciju, kā arī pievienojam konstruktorā vēl vienu parametru

public interface Passenger {  
 int passengerSeatCount();  
 }  
  
 public class Buss extends Car implements Passenger {  
  
 private final int passengerSeatCount;  
  
 public Buss(String color, int year, String manufacturer, int passengerSeatCount) {  
 super(color, year, manufacturer);  
 this.passengerSeatCount = passengerSeatCount;  
 }  
  
 @Override  
 public int passengerSeatCount() {  
 return this.passengerSeatCount;  
 }  
 }

* Klasē Car pievienojam geteru metodes

public class Car {  
  
 // . . .  
  
 public String getColor() {  
 return color;  
 }  
  
 public int getYear() {  
 return year;  
 }  
  
 public String getManufacturer() {  
 return manufacturer;  
 }  
}

* Katrā no klasēm pārrakstām toString metodi

public class Car {  
  
 // . . .  
  
 @Override  
 public String toString() {  
 return "Car{" +  
 "color='" + color + '\'' +  
 ", year='" + year + '\'' +  
 ", manufacturer='" + manufacturer + '\'' +  
 '}';  
 }  
 }  
  
 public class Buss extends Car implements Passenger {  
  
 // . . .  
  
 @Override  
 public String toString() {  
 return "Buss{" +  
 "color='" + getColor() + '\'' +  
 ", year='" + getYear() + '\'' +  
 ", manufacturer='" + getManufacturer() + '\'' +  
 ", passengerSeatCount=" + passengerSeatCount +  
 "} ";  
 }  
 }  
  
 public class Truck extends Car {  
  
 // . . .   
  
 @Override  
 public String toString() {  
 return "Truck{" +  
 "color='" + getColor() + '\'' +  
 ", year='" + getYear() + '\'' +  
 ", manufacturer='" + getManufacturer() + '\'' +  
 ", loadCapacity=" + loadCapacity +  
 "} ";  
 }  
 }

* Katrā no klasēm pārrakstām equals metodi

public class Car {  
  
 // . . .  
  
 @Override  
 public boolean equals(Object that) {  
 if (this == that) return true;  
 if (that == null || getClass() != that.getClass()) return false;  
 Car car = (Car) that;  
 return Objects.equals(color, car.color) &&  
 Objects.equals(year, car.year) &&  
 Objects.equals(manufacturer, car.manufacturer);  
 }  
 }  
  
 public class Buss extends Car implements Passenger {  
  
 // . . .  
  
 @Override  
 public boolean equals(Object that) {  
 if (this == that) return true;  
 if (that == null || getClass() != that.getClass()) return false;  
 Buss buss = (Buss) that;  
 return passengerSeatCount == buss.passengerSeatCount  
 && getColor().equals(buss.getColor())  
 && getYear() == buss.getYear()  
 && getManufacturer().equals(buss.getManufacturer());  
 }  
 }  
  
 public class Truck extends Car {  
  
 // . . .   
  
 @Override  
 public boolean equals(Object that) {  
 if (this == that) return true;  
 if (that == null || getClass() != that.getClass()) return false;  
 Truck truck = (Truck) that;  
 return loadCapacity == truck.loadCapacity  
 && getColor().equals(truck.getColor())  
 && getYear() == truck.getYear()  
 && getManufacturer().equals(truck.getManufacturer());  
 }  
 }

* Pārbaudīsim kā darbojas “equals” un “toString” metodes. Izveidjoiet jaunu klasi CarParking ar “main” metodi. Izveidosim divu objektus un salīdzināsim tos savā starpā un rezultātu izvadīsim konsolē.

public class CarParking {  
  
 public static void main(String[] args) {  
 Buss myNewBuss = new Buss("Red", 2020, "Volvo", 50);  
 Buss myOldBuss = new Buss("Yellow", 1990, "Man", 15);  
   
 System.out.println("myNewBuss: " + myNewBuss);  
 System.out.println("myOldBuss: " + myNewBuss);  
 System.out.println("Are busses equals: " + myNewBuss.equals(myOldBuss));  
 }  
   
 }

* Konsolē jabūt šādām rezultātam:

myNewBuss: Buss{color='Red', year='2020', manufacturer='Volvo', passengerSeatCount=50}   
myOldBuss: Buss{color='Red', year='2020', manufacturer='Volvo', passengerSeatCount=50}   
Are busses equals: false

* Nodemonstrēsim polimorfismu darbībā. Izveidosim vēl dažus objektus un ievietosim tos visus vienā sarakstā List un izdrukāsim tos.

public class CarParking {  
  
 public static void main(String[] args) {  
 // . . .  
  
 Truck myTruck = new Truck("black", 2010, "Man", 3000);  
 Car myCar = new Car("Green", 2021, "BMW");  
  
 List<Car> allMyVehicles = Arrays.asList(myNewBuss, myOldBuss, myTruck, myCar);  
 for (Car car : allMyVehicles) {  
 System.out.println(car);  
 }  
 }  
   
 }

* Konsolē jabūt šādām rezultātam:

Buss{color='Red', year='2020', manufacturer='Volvo', passengerSeatCount=50}   
Buss{color='Yellow', year='1990', manufacturer='Man', passengerSeatCount=15}   
Truck{color='black', year='2010', manufacturer='Man', loadCapacity=3000}   
Car{color='Green', year='2021', manufacturer='BMW'}

# 2. mājas darbs

# Klašu pārmantojamība un polimorfisms

### 

### **Noteikumi:**

Uzdevuma risināšanai izmantojam sekojošu modeli.

Graphical user interface, text

Description automatically generated

**Soļi:**

* Izveidojam interfeisu Payable un tajā vienu metodi “pay”
* Izveidojam abstraktu klasi Employee. Šī klase implementē interfeisu Payable. Šī klase ir abstrakta jo nav domāta tiešai izmantošanai, bet gan specializētu apakšklašu veidošanai. Šajā klasē ievietosim kopējās loģikas kodu un apakšklasēs tikai specializētu kodu.
  + Klasē ievietojam laukus: firstName, lastName, SSN (Social Security Number) un salary (alga)
  + Izveidojam 2 konstruktorus ar 3 un 4 parametriem, iekļaujam vienā no tiem salary.
  + Pievienojiet seteru un geteru metodes visiem laukiem
* Izveidojam specializētu Employee klasi nosaucot to HourlyEmployee.
  + Šajā klasē ievietojam laukus “int hoursWorked” un “double payRate”.
  + aizvietojam (override) metodi pay kura atgriež hoursWorked \* payRate. Pēc metodes izpildes hoursWorked = 0.
  + Pievienojam seteru un geteru metodes visiem laukiem
* Izveidojam specializētu Employee klasi nosaucot to SalariedEmployee.
  + Šajā klasē ievietojam laukus “double weeklySalary”.
  + aizvietojam (override) metodi pay kura atgriež weeklySalary.
  + Pievienojam seteru un geteru metodes visiem laukiem
* Izveidojam specializētu Employee klasi nosaucot to Executive.
  + Šajā klasē ievietojam lauku “double bonus” un metodi awardBonus(double), kas pieliek bonusu.
  + Metode “pay” atgriež virsklases pay metodes rezultātu ar pieskaitītu bonus. Pēc metodes izpildes bonus = 0.
  + Pievienojam seteru un geteru metodes visiem laukiem
* Izveidojam PayInterfaceApp klasi un tajā sekojošu main metodi

// create four-element Payable array  
 Payable[] payableObjects = new Payable[5];  
 //  
 payableObjects[0] =  
 new SalariedEmployee("John", "Smith", "111-11-1111", 800.00);  
 payableObjects[1] =  
 new SalariedEmployee("Lisa", "Barnes", "888-88-8888", 1200.00);  
 //  
 payableObjects[2] = new HourlyEmployee("Barn", "Nobles", "273-56-288", 15.0);  
 ((HourlyEmployee) payableObjects[2]).addHours(120);  
 payableObjects[3] = new HourlyEmployee("Scott", "Tiger", "123-13-789", 14.0);  
 ((HourlyEmployee) payableObjects[3]).addHours(144);  
 //  
 payableObjects[4] = new Executive("", "", "132-56-654", 1400d);  
 ((Executive) payableObjects[4]).awardBonus(1000);  
  
 for (Payable currentPayable : payableObjects) {  
 if (currentPayable != null) {  
 System.out.println("" +  
 currentPayable +  
 " payment due = " + currentPayable.pay());  
 }  
 }

* izpildot klasi parādāss aptuveni šāds rezultāts. Tajā redzams darbinieka vārds, uzvārds, ssn, algas komponentes un algas lielums.

> Task :PayableInterfaceTest.main()  
salaried employee: Employee John Smith 111-11-1111 weekly salary: $800.00 payment due = 800.0  
salaried employee: Employee Lisa Barnes 888-88-8888 weekly salary: $1,200.00 payment due = 1200.0  
hourly employee: Employee Barn Nobles 273-56-288 rate : $15.00 hoursWorkeed $120 payment due = 1800.0  
hourly employee: Employee Scott Tiger 123-13-789 rate : $14.00 hoursWorkeed $144 payment due = 2016.0  
executive employee: Employee 132-56-654 salary plus award = : $1,000.00 payment due = 2400.0