****

Latvijas Universitātes Vadības un uzņēmējdarbības mācību centrs

(LU VUMC)

**IEVADS JAVA PROGRAMMĒŠANĀ**

**NODARBĪBA #5**

Mārtiņš Ceske/Vjačeslavs Vasiļevskis

2021

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



# 1. mājas darbs

# Bankas karte

**Klases BankCard izveidošana**

* Izveidojam klasi bankas kartei ar nosaukumu BankCard
* Klasei būs konstruktors ar 6 parametriem:
  + Banka (bankName)
  + Maksājumu izpildītājs (origin)
  + Turētāja vārds (holderName)
  + Kartes numurs (number)
  + Derīguma termiņš (expiryDate)
  + Drošības kods (cvv)
* Izveidojam klasi "BankCard"

// Pievienojam "import" LocalDate bibliotēkai  
import java.time.LocalDate;  
  
 public class BankCard {  
  
 // Izveidojam mainīgos, kuri iekapsulē datus klasē.  
 // Dati nav tieši pieejami - izmanto "private" piekļves modifikātoru.  
 private String bankName;  
 private String origin;  
 private String holderName;  
 private String number;  
 private LocalDate expiryDate;  
 private String cvv;  
 private double balance;  
  
 // Izveidojam konstruktoru  
 public BankCard(String bankName,  
 String origin,  
 String holderName,  
 String number,  
 LocalDate expiryDate,  
 String cvv) {  
 this.bankName = bankName;  
 this.origin = origin;  
 this.holderName = holderName;  
 this.number = number;  
 this.expiryDate = expiryDate;  
 this.cvv = cvv;  
 this.balance = 0;  
 }  
  
 // Izveidojam metodi naudas izņemšanai  
 public void withdraw(double value) {  
 this.balance -= value;  
 }  
  
 // Izveidojam metodi naudas pievienošanai  
 public void addFunds(double value) {  
 this.balance += value;  
 }  
  
 }

**Klases BankCard objekta izveidošana**

* Izveidojam jaunu klasi CardUser, kuru izmantosim BankCard objekta veidošanai.
* Klasei CardUser pievienojam metodi “main”
* Metodē “main” izveidojam BankCard objektu

BankCard myCard = new BankCard(  
 "Banka Citadele",  
 "Visa",  
 "Roberts Polis",  
 "2345678345",  
 LocalDate.of(2024, 12, 10),  
 "134");

* Turpat metodē “main” arī izdrukājam kartes objektu

System.out.println("Mana karte: " + myCard);

* Kas sanāca? Tiek izdrukāts klases nosaukums un haškods pārvests heksadecimalājā sistēmā “class@ID”. Lai izdrukātu objekta laukus, mums ir nepieciešams pārrakstīt metodi “toString”

**Objekta datu izdrukāšana**

* Klasei BankCard pievienojam metodi “toString”

@Override  
 public String toString() {  
 return "BankCard{" +  
 "bankName='" + bankName + "' " +  
 ", origin='" + origin + "' " +  
 ", holderName='" + holderName + "' " +  
 ", number='" + number + "' " +  
 ", expiryDate=" + expiryDate +  
 ", cvv='" + cvv + "' " +  
 ", balance=" + balance +  
 '}';  
 }

* Izpildam klases CardUser metodi “main”. Kas tiek izdrukāts?

### **Objekta metožu izmantošana**

* Pievienosim kartei naudu. CardUser klasē metodē “main” izsauksim kartes objektam metodi “addFunds”

myCard.addFunds(100d);

* Izdrukājiet kartes saturu
* Izņemsim no kartes 20

myCard.withdraw(20d);

* Izdrukājiet saturu
* Izpildiet klases CardUser metodi “main”! Nospiediet uz zaļā stūrīša pie “class” vai “main”.

# 2. mājas darbs

# Bankas konts

### 

### **Noteikumi:**

* Izveidot klasi BankAccount ar laukiem:
  + “Konta numurs” ,
  + “Pin kods”,
  + “Debeta bilance”,
  + “Kredīta limits”,
  + “Izmantots kredīts”
* Klasei ir jānodrošina iespēja:
  + “Iemaksāt naudu” (topUp),
  + “Izņemt naudu” (whitDraw)
* Konta izmantošanas laikā nepieciešams padot pin kodu. Metodes topUp un withdraw pieņem ne tikai summu, bet arī pin kodu.

### **Prasības naudas izmaksai**

* Naudas izmaksa var tik veikta tikai tad, ja pin kods ir ievadīts pareizi
* Pirmkārt izmantot debeta bilanci
* Ja uz debeta konta nav naudas, tad atlikumā izmantojam kredīta bilanci.
* Izmantot kredīta kontu nedrīkst vairāk par kredīta līmeni (Piemērs #1)
* Kredīts nedrīkst pārsniegt kredīta limitu (Piemērs #2)
* PIN kodam ir jāievada obligāti
* Pirmkārt iemaksa nonāk kredīta kontā - kredīta segšanai, bet atlikusī summa tiek ieskaitīta debeta kontā (Piemērs #3)

### **Piemērs #1**

Debeta bilance = 100  
 Kredīta limits = 200  
 Izmantots kredīts = 0

* Izņemsim 150, pēc izmaksas konta stāvoklim jābūt tādam:

Debeta bilance = 0  
 Kredīta limits = 200  
 Izmantots kredīts = 50

* Debeta bilance tagad ir 0, tāpēc ka nepietika naudas uz konta, un tika izmantota kredīta bilance uz 50.

### **Piemērs #2**

Debeta bilance = 100  
 Kredīta limits = 200  
 Izmantots kredīts = 180

* Izņemsim 150, pēc izmaksas konta stāvoklim ir jābūt šādam:

Debeta bilance = 100  
 Kredīta limits = 200  
 Izmantots kredīts = 180

* Konta stāvoklis nemainās, jo debeta bilances nepietiek, un kredīta bilances atlikums ir tikai 20

### **Piemērs #3**

Debeta bilance = 100   
 Kredīta limits = 200   
 Kredīta aizņēmums = 180

* Iemaksājam 300, un konta stāvoklim ir jābūt šādam:

Debeta bilance = 220   
 Kredīta limits = 200   
 Kredīta aizdevums = 0

P.S.

* Simbolu virkņu salīdzināšanai savā starpā ir jāizmanto metode “equals”.
* Sākotnējā pārbaude naudas izņemšanai ir šāda:

public boolean topup(double amount, String pinCode){   
 //check pin code double availableAmount = debeta bilance + (kredītalimits - izmantots kredīts);   
 boolean isEnoughMoney = availableAmount - amount > 0.001   
 ....  
}