

Задачи - private

Точка

Да се надополни програмата со следните барања:

- да се креира класа на точка во тродимензионален простор и да се напише функција која ќе го пресметува растојанието помеѓу две такви точки.
- да се напише функција која како аргумент прима три точки во дводимензионален простор и ќе проверува дали тие точки лежат на иста права.

$\sqrt{(t1.x - t2.x)^2 + (t1.y - t2.y)^2}$ растојание

$\sqrt{(t1.x - t2.x)^2 + (t1.y - t2.y)^2 + (t1.z - t2.z)^2}$ растојание 3д

$(t2.y - t1.y) / (t2.x - t1.x) == (t3.y - t2.y) / (t3.x - t2.x)$ иста права

1 3 5 8	2.83 false
------------	---------------

Купувачка кошничка

Да се напише програма во која од стандарден влез се вчитува N (бројот на производи), а потоа се вчитуваат податоците за N производи (име, цена, количина). Програмата треба на стандарден излез да ја отпечати листата на купени производи и вкупната сума која треба да се плати во следниот облик (пример):

3 -----Vnesi artikl 1----- Vnesi ime na proizvod: Flips Vnesi cena:10 Vnesi kolicina: 3 CocaCola 75 2 ChokoBanana 5 10	1. Flips 10.00 x 3 = 30.00 2. CocaCola 75.00 x 2 = 150.00 3. ChokoBanana 5.00 x 10 = 50.00 Total: 230.00
--	---

Агол

Да се дефинира класа `Agol`, во која се чуваат информации за:

- степени, минути и секунди (int)

Во класата да се реализираат:

- конструктор по потреба
- методи за поставување на вредности на атрибутите на класата (set методи)
- метод за пресметување на вредноста на аголот во секунди

Да се дефинира и метод за проверување на тоа дали внесениот агол е валиден, односно дали се внесени соодветни вредности за атрибутите (во границите кои ги дозволуваат).

`(step >= 0 && step <= 360) && (minu >= 0 && minu <= 60) && (sek >= 0 && sek < 60)`

180 0 60	Nevalidni vrednosti za agol
----------	-----------------------------

Круг

Да се дефинира класа `Krug`, во која се чуваат информации за:

- радиус `float`
- бројот π `const float`.

Во класата да се реализираат:

- default конструктор и конструктор со аргументи
- метод за пресметување плоштина
- метод за пресметување периметар
- метод кој кажува дали плоштината и периметарот на даден круг се еднакви

$\pi \cdot \text{radius} \cdot \text{radius}$ - плоштина

$2 \cdot \text{radius} \cdot \pi$ - периметар

5	31.4 78.5 false
---	-----------------------

Филм

Да се дефинира класа `Movie`, во која ќе се чуваат информации за:

- име
- режисер
- жанр
- година

Сите променливи треба да бидат приватни. Соодветно во рамките на класата да се дефинираат:

- default конструктор и конструктор со аргументи
- метод за печатење на информациите за филмот

Дополнително да се реализира надворешна функција:

- `void pecati_po_godina(List<Movie> movies, int godina)` која ќе прима аргумент низа од филмови, вкупниот број на филмови и година, а треба да ги отпечати само филмовите кои се направени во дадената година.

4	Ime: Lincoln Reziser: Steven_Spielberg Zanr: History Godina: 2012
Frankenweenie Tim_Burton Animation 2012	Ime: Frankenweenie Reziser: Tim_Burton Zanr: Animation Godina: 2012
Lincoln Steven_Spielberg History 2012	
Wall-E Andrew_Stanton Animation 2008	
Avatar James_Cameron Fantasy 2009	
2012	

Уредување на дом

Во оваа задача е потребно да уредите даден дом со маси. Креирајте класа `Masa` со следниве атрибути:

- должина (целобројна вредност)
- ширина (целобројна вредност)

конструктор со и без параметри и метода `pecati()`.

Креирајте класа `Soba` која содржи:

- маса (објект од класата `Masa`)
- должина на собата (целобројна вредност)
- ширина на собата (целобројна вредност)

конструктор со и без параметри и метода `pecati()` во која се повикува и `pecati()` за објектот `Masa`.

Креирајте класа `Kukja` со атрибути:

- соба (објект од класата `Soba`)
- адреса (низа од 50 знаци), и соодветни методи.

конструктор со и без параметри и метода `pecati()` во која се повикува и `pecati()` за објектот `Soba`.

3 2 4 10 20 Goce_Delcev_20 1 1 12 43 Pitu_Guli_2 2 4 10 20 Partizanski_Odredi_87_b	Adresa: Goce_Delcev_20 Soba: 10 20 Masa: 2 4 Adresa: Pitu_Guli_2 Soba: 12 43 Masa: 1 1 Adresa: Partizanski_Odredi_87_b Soba: 10 20 Masa: 2 4
---	---

Договор

Да се дефинира класа `Potpisuvac` во која се чуваат информации за:

- име
- презиме
- ЕМБГ

За класата да се дефинира default конструктор и конструктор со аргументи.

Да се дефинира класа `Dogovor`, во која се чуваат информации за:

- број на договор (int),
- категорија на договор (низа од 50 знаци),
- поле од 3 потпишувачи на договорот (објекти од класата `Potpisuvac`)

Во класата да се додаде метод кој ќе проверува дали постојат два исти потпишувачи (имаат ист ЕМБГ).

<pre>1 0101988450001 Alek Aleksov 0101988450001 Alek Aleksov 0202987440022 Marko Markov 1 Osiguruvanje</pre>	<pre>Dogovor 1: Postojat potpishuvaci so ist EMBG</pre>
--	---

Фабрика

Креирајте класа `Rabotnik` која во себе содржи:

- `Ime`
- `Prezime`
- `Plata`

За оваа класа да се креираат **default конструктор** и **конструктор** со аргументи. Да се имплементираат и следните методи:

- `GetPlata()` која ја враќа платата на работникот
- `Pecati()` која ги печати името, презимето и платата.

Креирајте класа `Fabrika` во која има:

- `rabotnik` (низа од вработени)
- `brojVraboteni` (целобројна вредност)

Во класата имплементирајте ги следните методи:

- `PecatiVraboteni()` ги печати сите вработени
- `PecatiSoPlata(int plata)` ги печати сите вработени со плата поголема или еднаква на дадената во аргументот `(int plata)`.

Во главната функција се внесуваат податоци за `n` вработени. Притоа прво се внесува `n`, па податоците за сите `n` вработени. Во последниот ред се чита минималната плата.

На излез да се прикажат прво сите вработени, а потоа само оние со поголема плата од минималната. Треба да се корисатат методите `PecatiVraboteni` и `PecatiSoPlata!`

3 Mile Palkovski 20000 Kalina Saleska 40720 Aco Noveski 66320 30000	Site vraboteni: Mile Palkovski 20000 Kalina Saleska 40720 Aco Noveski 66320 Vraboteni so plata povisoka od 30000 : Kalina Saleska 40720 Aco Noveski 66320
---	---

Насловна страница

Во оваа задача потребно е да се внесат податоци за насловна страница на списание.

За претставување на насловната страница напишете класа `FrontPage` која ќе содржи:

- објект од класата `NewsArticle` која ја претставува насловната вест на страницата
- цена (`float price`) со предодредена вредност 0
- број на издание на списанието (`int editionNumber`) со предодредена вредност 0

За класата `FrontPage` напишете предодреден (default) конструктор и конструктор со параметри.

Класата `NewsArticle` треба да содржи:

- објект од класата `Category` која ја претставува категоријата во која спаѓа веста
- наслов од максимум 30 знаци со предодредена вредност `untitled`

За класата `NewsArticle` напишете предодреден конструктор и конструктор со параметри.

Класата `Category` треба да содржи име од максимум 20 знаци (`char name[20]`) со предодредена вредност `unnamed`.

За сите класи треба да напишете соодветен метод за печатење `print()`.

За категоријата се печати само името: `Category: [name]`.

За веста се печати насловот, па категоријата во нов ред:

```
Title: [title]
category.print()
```

За насловната страница се печати цената и изданието во прв ред, па веста во втор:

```
Price: [price], Edition number: [editionNumber]
article.print()
```

Автомобил

Во оваа задача треба да се внесат и испечатат податоци за автомобили.

За еден автомобил (објект од класата `Car`) се чуваат следниве податоци:

- сопственик (објект од класата `Person`)
- датум на купување (објект од класата `Date`)
- цена (`float price`), предодредена вредност 0

За класата `Car` потребно е да се напише метод за печатење `print()` и метод за добивање на цената `getPrice()`.

Класата `Date` содржи три цели броеви (`int year, month, day`) кои претставуваат датум. За неа треба да се напише метод за печатење `print()`, предодреден (default) конструктор, конструктор со параметри и конструктор за копирање.

Класата `Person` содржи име и презиме (низи од максимум 20 знаци, со предодредени вредности `not specified`), предодреден конструктор, конструктор со параметри и метод за печатење `print()`.

Методот за печатење кај класата `Person` изгледа вака: `[name] [lastName]`.

Методот за печатење кај класата `Date` изгледа вака: `[year].[month].[day]`.

Методот за печатење кај класата `Car` изгледа вака:

```
owner.print()
```

```
date.print()
```

```
Price: [price]
```

Покрај тоа, потребно е да се напише метод `cheaperThan(List<Car> cars, int numCars, float price)` кој ќе ги испечати сите објекти `Car` од низата `cars` со големина `numCars` чија цена е помала од `price`.

Riste Risteski 2019 12 12 230000	Riste Risteski 2019.12.12 Price: 230000
---	---

Сончев систем

Креирајте класа `Planeta` која во себе содржи:

- `ime` (низа од максимум 30 знаци);
- `dijametar` (целобројна вредност);
- `oddalechenost` (целобројна вредност);
- `masa` (децимален број).

За оваа класа да се креираат **default конструктор**, **конструктор со аргументи** и **сору конструктор**. Да се имплементираат потребните `get` и `set` методи, како и следната метода:

- `pechati()` - ги печати името и дијаметарот на планетата, во формат: "[ime] so dijametar [dijametar]km.\n"

Креирајте класа `SonchevSistem` која содржи:

- `najbliskaPlaneta` (објект од класата `Planeta`)
- `planeti[50]` (низа од планети)
- `brojPlaneti` (целобројна вредност)

Во класата креирајте го потребниот конструктор и имплементирајте ги следните методи:

- `pechatiPlaneti()` најпрво ги печати сите планети, а потоа ја печати најблиската планета во формат: "Najbliska planeta e [ime] so dijametar [dijametar]km.\n";
- `pechatiSoMasa(float masa)` ги печати сите планети со маса поголема од дадената во аргументот `(float masa)`.

При печатењето треба да ја искористите методата `pechati()` од `Planeta`. Најблиска планета до сонцето е онаа чија `oddalechenost` е најмала.

```
6
Merkur 4879 57909227 0.0553
Venera 12104 108209475 0.815
Zemja 12742 149598262 1
Mars 6779 227943824 0.11
Jupiter 139822 778340821 317.8
Saturn 116464 1426666422 95.2
1
```

```
Site planeti:
Merkur so dijametar 4879km.
Venera so dijametar 12104km.
Zemja so dijametar 12742km.
Mars so dijametar 6779km.
Jupiter so dijametar 139822km.
Saturn so dijametar 116464km.
Najbliska planeta e Merkur so
dijametar 4879km.
```

```
Planeti so masa pogolema od 1kg:
Jupiter so dijametar 139822km.
Saturn so dijametar 116464km.
```

Недвижнина

Да се развие класа `Nedviznina` за која се чуваат следниве информации:

- адреса (динамички алоцирана низа од знаци)
- квадратура (цел бој)
- цена за квадрат (цел бој)

За оваа класа да се имплементираат соодветните конструктори и следните методи:

- `cena()` кој ќе ја враќа цената на недвижнината (квадратура * цена-за-квадрат)
- `pecati()` кој ќе ги испечати сите информации за истата
- `danokNaImot()` кој го ваќа данокот што се плаќа за недвижнината, а истиот се пресметува како 5% од цената на недвижнината.

Од оваа класа да се изведе класа `Vila` за која дополнително се чува данок на луксуз (цел бој, пр. 10%). За оваа класа да се преоптоварат методите:

- `pecati()`
- `danokNaImot()` со тоа што пресметаниот данок се зголемува процентуално за данокот на луксуз.

<pre>Kukja_vo_Centar 60 850 Vila_na_Vodno 110 1120 10</pre>	<pre>Kukja_vo_Centar, Kvadratura: 60, Cena po Kvadrat: 850 Danok za: Kukja_vo_Centar, e: 2550 Vila_na_Vodno, Kvadratura: 110, Cena po Kvadrat: 1120, Danok na luksuz: 10 Danok za: Vila_na_Vodno, e: 18480</pre>
---	--

Спортски екипи

Да се дефинира класа `Ekipa` за која се чуваат следниве информации:

- името на екипата (низа од најмногу 15 знаци)
- број на порази
- број на победи

За оваа класа да се дефинира метод `pecati()` која ги печати податоците за екипата. Од оваа класа да се изведе нова класа, `FudbalskaEkipa`.

За фудбалската екипа дополнително се чуваат информации за:

- вкупниот број на црвени картони
- вкупниот број жолти картони
- бројот на нерешени натпревари

За фудбалската екипа да се преоптовари методот `pecati()`, така што покрај останатите информации, ќе се испечатат и бројот на нерешени резултати и вкупен број на поени во формат: Име на екипа, број на победи, број на порази, број на нерешени натпревари и вкупен број на поени (за победа фудбалската екипа добива 3 поени, додека за нерешен резултат, 1 поен);

<code>Fudbaleri</code> 5 4 3 7 6	<code>Ime: Fudbaleri Pobedi: 5 Porazi: 4</code> <code>Nereseni: 6 Poeni: 21</code>
---	---

Мој Термин

Со цел да се подобри системот Мој Термин, со воведување функционалност за пресметување плати за лекарите за еден месец, од Министерството за здравство на Република Македонија, ги добивате следните задачи:

Да се креира класа `Lekar` во која што ќе се чуваат:

- факсимил на докторот (цел број)
- име (низа од максимум 10 знаци)
- презиме (низа од максимум 10 знаци)
- почетна плата (децимален број)

За класата да се имплементираат методите:

- `void pecati()`: Печати информации за лекарот во формат `Факсимил: име презиме`
- `double plata()`: ја враќа платата на лекарот

Да се креира класа `MaticenLekar` која што наследува од `Lekar` и во неа се чуваат дополнителни информации за:

- број на пациенти со којшто лекарот соработувал во текот на месецот (цел број)
- котизации наплатени од пациентите во текот на месецот (динамички алоцирана низа од децимални броеви)

За класата да се препокријат методите:

- `void pecati()`: ги печати основните информации за лекарот, а во нов ред го печати и просекот од наплатените котизации
- `double plata()`: ја враќа платата на матичниот лекар
 - Платата на матичниот лекар се пресметува со зголемување на основната плата за 30% од просекот од наплатените котизации за месецот

<pre>7 766433 Cvetanka Cvetkova 27899.90 123122 Stefan Stefanov 31789.50 454323 Trajce Trajkov 19458.30 343989 Goran Trajkov 28945.10 515788 Nikola Nikolov 36985.50 784512 Viktorija Stojanovska 37855.00 985623 Ivana Ivanova 38745.70 5 1000 2000 2500 7800 5560 4 1000 2000 3000 10000 6 7800 7800 8000 9000 900 1000 5 1000 1500 2000 2300 2400 3 15000 10000 7580 4 10000 7000 8000 1000 3 1000 2000 3000 1 1</pre>	<pre>===TESTIRANJE NA KLASATA LEKAR=== 766433: Cvetanka Cvetkova Osnovnata plata na gorenavedeniot lekar e: 27899.9 123122: Stefan Stefanov Osnovnata plata na gorenavedeniot lekar e: 31789.5 454323: Trajce Trajkov Osnovnata plata na gorenavedeniot lekar e: 19458.3 343989: Goran Trajkov Osnovnata plata na gorenavedeniot lekar e: 28945.1 515788: Nikola Nikolov Osnovnata plata na gorenavedeniot lekar e: 36985.5 784512: Viktorija Stojanovska Osnovnata plata na gorenavedeniot lekar e: 37855 985623: Ivana Ivanova Osnovnata plata na gorenavedeniot lekar e: 38745.7</pre>
---	---

Employee

Да се дефинира апстрактна класа `Employee` којашто ќе содржи име на вработениот, години и работно искуство во години (`integer`). Да се дефинираат чисти виртуелни функции `plata()` и `bonus()` (`double`).

Од класата `Employee` да се изведе класа `SalaryEmployee` која покрај основните информации содржи и информација за основната плата. Бонусот на овие работници се пресметува како процент од основната плата, а процентот е бројот на години со работно искуство. На пример ако работеле 10 години, бонусот е 10 проценти од основната плата. Вкупната плата се пресметува како основната плата плус бонусот.

Од класата `Employee` исто така да се изведе класа `HourlyEmployee` која покрај основните информации содржи информација и за вкупниот број на часови кои ги одработил работникот и платата по час. Вкупната плата се пресметува како бројот на часови помножен со платата по час плус бонусот, додека бонусот се пресметува на следниот начин: За секој час над 320-тиот се добива 50 проценти од платата по час.

Од класата `Employee` на крај се изведува класата `Freelancer` која покрај основните информации содржи и број на проекти на кои работел вработениот и низа со суми кои ги добил за тие проекти (`double`). По направени 5 проекти, за секој нареден вработените добиваат бонус од 1000 денари. Вкупната плата на овој тип на вработени изнесува вкупната сума добиена од сите проекти плус бонусот.

Да се напише функција кој ќе прима два објекти од класата `Employee` и ќе ги споредува според тоа дали имаат ист број на години и дали добиваат ист бонус.

Да се дефинира класа `Company` која ќе содржи информации за името на компанијата, бројот на вработени, и низа од класата `Employee` или `Employeec`. За потребите на оваа класа треба да се дефинира конструктор кој прима само еден аргумент - името на компанијата

- `double vkupnaPlata()` - метод којшто ја враќа вкупната плата на сите вработени во компанијата
- `double filtriranaPlata(List<Employee> emp)` - метод којшто ја враќа платата само на работниците кои се еднакви со дадениот вработен (според оператор `==`)
- `void pecatiRabotnici()` - метод којшто печати по колку вработени има од секој тип на работници во компанијата, а форматот на печатење можете да го видите од тест примерите

```
Smetkovodstveno_biro_Ekonomik
7
1 Krume_Petrov 44 11 27000
1 Liljana_Dimovska 47 16 24000
1 Petar_Ristov 34 8 18000
2 Tea_Vinarova 29 4 330 60
2 Mitko_Drenkovski 51 20 280 75
3 Iva_Damjanovska 37 14 6 3000 3300 2900 3450 1980 4010
3 Marko_Timov 24 1 3 2500 2700 2700
```

```
Vo kompanijata Smetkovodstveno_biro_Ekonomik rabotat:
Salary employees: 3
Hourly employees: 2
Freelancers: 2
Vkupnata plata e: 145890
Filtrirana plata e: 0
```

Игротека

Во една игротека има 2 типа играчки: **топки и коцки**. Коцките и топките се опишани со параметри како што се:

- боја (string)
- густина (int).

Дополнително за топките се знае и радиусот (int), додека за коцките целосните димензии (висина, ширина и длабочина – int).

За секоја од класите треба да се дефинираат методи `getMasa()` и `getVolumen()`. Масата на играчката се пресметува како производ од волуменот и густината на играчката. За PI користете ја вредноста 3.14.

Во функцијата `main` да се декларира променлива `kupche` што претставува динамички алоцирана низа кон `Igrachka`. Во зависност од првиот влезен параметар се внесуваат објекти од класите `Торка` или `Коска` (1 - се внесува торка, 2 - се внесува коцка).

Од тастатура да се внесат податоци за коцката на Петра `Kocka petra`. Во главната функција во да се отпечатат:

1. Да се отпечати DA ако вкупната маса на сите играчки е поголема од масата на играчката на Петра, а NE во спротивно.
2. Разликата по апсолутна вредност на волуменот на играчката со максимален волумен во купчето и волуменот на коцката на Петра. Форматот е:
Razlikata e: {razlika}

Задачата да се реши со тоа што класите `Kocka` и `Topka` ќе наследуваат од класите `Forma` и `Igrachka`.

Дополнителни барања:

1. Во класата `Igrachka` да се додаде уште една чисто виртуелна функција `float getPlostina()`. Истата да се имплементира во класите `Kocka` и `Topka`
2. Во главната функција, дополнително да се испечати и: Разликата по апсолутна вредност на плоштината на играчката со минимална плоштина во купчето и плоштината на коцката на Петра во истиот формат како и второто барање погоре.

<pre>3 1 zelena 50 7 2 crvena 10 2 3 1 2 zolta 10 1 2 3 zelena 15 2</pre>	<pre>DA Razlikata e: 1424.03 Razlikata e: 10</pre>
---	--

30	
----	--

SocialNetwork

Да се дефинира апстрактна класа `User` за која ќе се чуваат:

- username (char[50])
- password (char[50])
- email (char[50])

Класата треба да содржи еден чист виртуелен метод `double popularity()`.

Од оваа класа да се изведат две класи и тоа `FacebookUser` и `TwitterUser`.

За класата `FacebookUser` уште се чуваат и:

- број на пријатели
- број на лајкови и
- број на коментари

Популарноста се пресметува така што се собираат бројот на пријатели, бројот на лајкови и бројот на коментари, но притоа бројот на лајкови се множи со коефициент на вредност на лајкот кој е ист за секој корисник и изнесува `0.1`, а исто така и бројот на коментари се множи со ист таков коефициент кој е `0.5`.

За класата `TwitterUser` уште се чуваат и:

- број на следачи и
- број на твитови

Популарноста се пресметува така што се собираат бројот на следачи и бројот на твитови, но притоа бројот на твитови се множи со коефициент на вредност на твитот кој е ист за секој корисник и изнесува `0.5`.

Да се креира класа `SocialNetwork` која ќе содржи:

- динамичка низа од покажувачи од класата `User`
- број на тековни корисници и
- максимален број на корисници кој може да го содржи мрежата и кој е ист за сите мрежи и иницијално е поставен на `5`

Да се дефинира метод `avgPopularity()` кој ќе ја враќа просечната популарност на корисниците во мрежата.

Исто така да се овозможи промена на максималната големина на низата преку методот `changeMaximumSize(int number)`.

Потребно е да се справите со следните исклучоци:

1. Доколку лозинката на корисникот не содржи барем 1 голема буква, 1 мала буква и 1 број да се фрли исклучок од класа `InvalidPassword` така што како параметар ќе се прати пораката `Password is too weak.`
2. Доколку емаилот на корисникот не содржи точно еднаш `@` да се фрли исклучок од класа `InvalidEmail` така што како параметар ќе се прати пораката `Mail is not valid.`
3. Доколку проба да се додаде корисник во социјалната мрежа, а веќе максималниот број на корисници е пополнет да се фрли исклучок од класа `MaximumSizeLimit` така што како параметар ќе се прати максималниот број на корисници во мрежата.

Сите класи кои се справуваат со исклучоци треба да го имаат имплементирано методот `void message()` така што за првите две класи ќе ја печати пораката којашто е испратена како параметар, а за последната класа ќе печати `You can't add more than N users`, каде што N е параметарот кој е пратен. Исто така со `try-catch` блокови справете се со исклучоците на соодветните места во `main()`, каде што во `catch` ќе го повикате методот `message()` од соодветниот исклучок.

<pre>4 blazer Gargamel2 blazer@yahoo.com 1 123 411 204 Scooby cart00nNetw0rk scoobydoo@gmail.com 1 282 1098 41 IronMan Avang3rs iron@man.com 2 678 1025 Dexter Massuc0 lisbon@dexter.com 2 418 299 EdSheeran Dlvld3 edsheeran@sheeran.com 2 10423 188</pre>	1529.98
---	---------

Трансакции

Да се креира класа `Transakcija` во која што ќе се чуваат информации за:

- датумот на реализирање на банкарската трансакција:
 - ден (`int`)
 - месец (`int`)
 - година (`int`)
- паричниот износ кој се однесува на трансакцијата (позитивен или негативен, тип `double`)
- моменталната вредност на еврото во денари (`static double EUR`), иницијално поставен на 61
- моменталната вредност на доларот во денари (`static double USD`), иницијално поставен на 50

За класата да се имплементираат соодветните конструктори, како и да се дефинираат следните четири чисто виртуелни методи:

- `double voDenari()`
- `double voEvra()`
- `double voDolari()`
- `void pecati()`

Трансакциите можат да бидат **денарски** и **девизни** (`DenarskaTransakcija` и `DeviznaTransakcija`). За девизните трансакции се чува дополнителна информација за валутата на трансакцијата (низа од три знаци). Дозволен валути за девизните трансакции се `USD` и `EUR`.

За двете изведени класи да се напишат соодветните конструктори и да се препокријат потребните методи.

Да се дефинира класа `Smetka` во која што ќе се чуваат информации за:

- извршените трансакции (динамички алоцирана низа од покажувачи кон класата `Transakcija`)
- број на извршените трансакции (`int`)
- број на сметката (низа од 15 знаци)
- почетно салдо во денари (`double`)

За класата `Smetka` да се имплементираат:

- потребен конструктор (со два аргументи, видете во `main`)
- `void izvestajVoDenari()` - функција што печати информации за сметката во форматот: `Korisnikot so smetka: [број на сметката] ima momentalno saldo od [салдо на сметката пресметано во денари] MKD`
- `void izvestajVoEvra()` - функција што печати информации за сметката во форматот: `Korisnikot so smetka: [број на сметката] ima momentalno saldo od [салдо на сметката пресметано во евра] EUR`
- `void izvestajVoDolari()` - функција што печати информации за сметката во форматот: `Korisnikot so smetka: [број на сметката] ima momentalno saldo od [салдо на сметката пресметано во долари] USD`
- `void pecatiTransakcii()` - функција што ги печати сите внесени трансакции

Да се креираат класи за следните исклучоци:

- `InvalidDateException` којшто се фрла доколку при креирање на трансакција не се испочитувани правилата `1<=ден<=31` и `1<=месец<=12`
- `NotSupportedCurrencyException` којшто се фрла доколку при креирање на девизна трансакција се внесува вредност за валута што не е дозволена

Овие исклучоци да се фрлат и да се фатат таму каде што е потребно. Истите при фаќање треба да печатат пораки од следниот формат:

- `Invalid Date 32/12/2018`
- `GBP is not a supported currency`

5	===VNESUVANJE NA TRANSAKCIITE I SPRAVUVANJE SO ISKLUCOCI=== GBP is not a supported currency Invalid Date 22/14/2018 ===PECHATENJE NA SITE TRANSAKCII=== 20/4/2018 1857.55 MKD 21/4/2018 1000 EUR 22/4/2018 13155.5 MKD ===IZVESHTAJ ZA SOSTOJBATA NA SMETKATA VO DENARI=== Korisnikot so smetka: 300047024112789 ima momentalno saldo od 77513.1 MKD ===IZVESHTAJ ZA SOSTOJBATA NA SMETKATA VO EVRA=== Korisnikot so smetka: 300047024112789 ima momentalno saldo od 1270.71 EUR ===IZVESHTAJ ZA SOSTOJBATA NA SMETKATA VO DOLARI=== Korisnikot so smetka: 300047024112789 ima momentalno saldo od 1550.26 USD ===PROMENA NA KURSOT NA EVROTO I DOLAROT=== ===IZVESHTAJ ZA SOSTOJBATA NA SMETKATA VO DENARI=== Korisnikot so smetka: 300047024112789 ima momentalno saldo od 78613.1 MKD ===IZVESHTAJ ZA SOSTOJBATA NA SMETKATA VO EVRA=== Korisnikot so smetka: 300047024112789 ima momentalno saldo od 1265.91 EUR ===IZVESHTAJ ZA SOSTOJBATA NA SMETKATA VO DOLARI=== Korisnikot so smetka: 300047024112789 ima momentalno saldo od 1578.58 USD
1 20 04 2018 1857.55	
2 21 04 2018 1234.55 GBP	
2 21 04 2018 1000 EUR	
1 22 14 2018 1200	
1 22 04 2018 13155.50	
62.1 49.8	