Objektno orijentirano programiranje
Međuispit
(24.04.2019.)

Ispit nosi ukupno **25 bodova** i piše se **150 minuta**. **Zadatke 1 i 4 potrebno je riješiti na ispitu u prostoru predviđenom za pisanje koda, dok je zadatke 2 i 3 potrebno riješiti na unutrašnjosti košuljice.**
U zadacima nije potrebno pisati dio u kojem se uključuju klase ili paketi klasa (import).

1. ZADATAK (8 bodova)

a) U prvom zadatku potrebno je modelirati klase i sučelja koja su nužna za opis pojednostavljenog modela glazbene industrije. Na koncertima (`Concert`) ili festivalima (`Festival`) nastupaju razni bendovi (`Band`) koji su karakterizirani svojim imenom te članovima, odnosno glazbenicima (`Musician`).

Kod u nastavku je potrebno nadopuniti na način da se omogući potpuna funkcionalnost klase `Main`, priložene u nastavku. Pretpostavite kako su svi `getteri` i `setteri` (za one varijable koje se mogu mijenjati) već napisani te ih ne trebate pisati, kao i da su metode `equals` nadjačane gdje trebaju biti.

NAPOMENA: u zadatku je potrebno koristiti konstruktor nadklase kada god je to moguće te poštivati dobru praksu objektno orijentirane paradigme.

```
public class Main {

	public static void main(String[] args) {

		Musician m1 = new Musician("Dave Grohl", "USA", 50,
				BandPosition.SINGER, 98);
		Musician m2 = new Musician("Pat Smear", "USA", 59,
				BandPosition.GUITARIST, 85);
		Musician m3 = new Musician("Chris Shiflett", "USA", 47,
				BandPosition.GUITARIST, 85);
		Musician m4 = new Musician("Nate Mendel", "USA", 50,
				BandPosition.BASSIST, 93);
		Musician m5 = new Musician("Taylor Hawkins", "USA", 47,
				BandPosition.SINGER, 70);
		Musician m6 = new Musician("Rami Jaffee", "USA", 50,
				BandPosition.PIANIST, 72);

		Musician[] musicians1 = new Musician[] {m1,m2,m3,m4,m5,m6};

		Band b1 = new Band(musicians1, "Foo Fighters");

		Manager manager = new Manager("Steven Howard", "USA", 67, b1);
		System.out.println(b1);

		manager.kickBandMember(m6);
		System.out.println(b1);
```

```
            System.out.println("Band member " + m3.getName() + " before training: "
                        + "Age: " + m3.getAge() + " Skill: " + m3.getSkill());

            m3.trainForOneYear();
            System.out.println("Band member " + m3.getName() + " after training: "
                        + "Age: " + m3.getAge() + " Skill: " + m3.getSkill());

            Event c1 = new Concert("PulaConcert", "Croatia",
                        "19 June",
                        b1, new String[] {"Best of You", "My Hero",
                                    "Big Me"});
            System.out.println("----CONCERT POSTER----");

            ((EventPromoter) c1).printEventPoster();

            Musician m7 = new Musician("Dan Auerbach", "USA", 39,
                        BandPosition.SINGER, 98);
            Musician m8 = new Musician("Patrick James Carney", "USA", 38,
                        BandPosition.DRUMMER, 97);

            Musician[] musicians2 = new Musician[] {m7, m8};

            Band b2 = new Band(musicians2, "The Black Keys");

            Person m9 = new Musician("Jack Black", "USA", 49,
                        BandPosition.SINGER, 100);
            Person m10 = new Musician("Kyle Gass", "USA", 58,
                        BandPosition.GUITARIST, 100);

            Person p1 = new Person("Goran Bare", "Croatia", 53);

            Band b3 = new Band(new Musician[] {(Musician) m9,  (Musician) m10},
                        "Tenacious D");

            Band[] bands = new Band[] {b1, b2, b3};

            Event f1 = new Festival("Rock in Croatia", "Zagreb",
                        "5 June", bands);
            System.out.println("----FESTIVAL POSTER----");
            ((EventPromoter) f1).printEventPoster();

      }
}
```

ISPIS

```
Members of the band Foo Fighters are: Dave Grohl, Pat Smear, Chris Shiflett, Nate Mendel,
Taylor Hawkins, Rami Jaffee
Members of the band Foo Fighters are: Dave Grohl, Pat Smear, Chris Shiflett, Nate Mendel,
Taylor Hawkins
Band member Chris Shiflett before training: Age: 47 Skill: 85
Band member Chris Shiflett after training: Age: 48 Skill: 86
----CONCERT POSTER----
Event date: 19 June
Band: Foo Fighters
```

(NASTAVAK ISPISA JE NA SLJEDEĆOJ STRANICI)

```
Song list for this concert is:
Song 1: Best of You
Song 2: My Hero
Song 3: Big Me
----FESTIVAL POSTER----
Event date: 5 June
Festival: Rock in Croatia
Band list for the festival is:
Band 1: Foo Fighters
Band 2: The Black Keys
Band 3: Tenacious D
```

Svaki glazbenik u bendu ima ulogu (`BandPosition`) koja je definirana konačnim skupom uloga te određenu razinu vještine za tu ulogu. Nadalje, svaki glazbenik kroz svoju karijeru može **treningom napredovati** na način da se za **godinu dana njegov `skill` uveća za jedan** (čime očigledno postaje stariji za jednu godinu). Svaki član benda, osim uloge i pripadajuće vještine ima i sljedeće vrijednosti: svoje ime, državu iz koje dolazi te određen broj godina. U glazbenoj industriji, osim glazbenika postoji i vrlo bitan entitet menadžera (`Manager`). Kao i glazbenik, on ima svoje ime, državu iz koje dolazi te broj godina. Osim prethodno navedenih karakteristika, **menadžer može izbaciti jednog od članova benda** (potrebno implementirati na način da se polje članova benda umanji te izbaci odgovarajući član – duljina polja mora odgovarati broju članova benda; nije potrebno provjeravati nalazi li se član benda u polju, možete pretpostaviti da se nalazi). **Osigurajte** da se objekti instancirani iz klase `Band` **mogu ispisati** na način opisan u metodi `main`.

```
public ____________________ Person {

	private final String name;
	private final String country;
	private int age;









}
```

```
public enum BandPosition {
	 SINGER, GUITARIST, BASSIST, DRUMMER, PIANIST
}
```

```
public ____________________Musician ________________________________________________{

	private Integer skill;
	private BandPosition bandPosition;









	public void trainForOneYear() {









	}

}
```

```
public _____________ Band {

	private Musician[] bandMembers;
	private String bandName;









}
```

```
public ____________________ Manager ____________________________________________{

	private Band managingBand;









	public void kickBandMember(Musician musician) {

	}
}
```

U sklopu ovog pojednostavljenog modela, definirali smo dva događaja (`Event`, klasa koju nije moguće instancirati): koncert i festival. Oba događaja imaju atribute ime, lokaciju te datum održavanja. **Koncert kao događaj je definiran bendom koji nastupa te popisom pjesama koje taj bend izvodi, dok je festival definiran popisom bendova koji nastupaju na tom festivalu.** Oba događaja imaju funkcionalnost ispisa postera koja je specificirana sučeljem `EventPromoter`. Poster za koncert sadrži informaciju o datumu događaja, bendu koji nastupa te popis pjesama, a poster za festival sadrži datum događaja, ime festivala te popis bendova koji na tom festivalu nastupaju.

```
public _____________________________ Event {

	private final String name;
	private final String location;
	private String date;









}
```

```
public interface EventPromoter {

	public void printEventPoster();

}
```

```
public _____ Festival _____ {

	private Band[] bandList;









}
```

```
public _____ Concert _____{

	private Band playingband;
	private String[] songList;









}
```

b) Dva se glazbenika (isključivo glazbenika) međusobno mogu i uspoređivati po svojoj vještini (pretpostavite da se svaka pozicija u bendu vrednuje jednako, npr. gitarist sa vještinom 100 je jednak pjevaču sa vještinom 100). Napišite klasu `SkillComparator` koja implementira navedenu funkcionalnost putem sučelja `Comparator<T>`.

```
public _____ SkillComparator _____{




}
```

Interface Comparator<T>

Modifier and Type	Method and Description
int	**compare**(T o1, T o2) Compares its two arguments for order.

Method Detail

compare

```
int compare(T o1,
            T o2)
```

Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Parameters:

o1 - the first object to be compared.

o2 - the second object to be compared.

Returns:

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Throws:

NullPointerException - if an argument is null and this comparator does not permit null arguments

ClassCastException - if the arguments' types prevent them from being compared by this comparator.

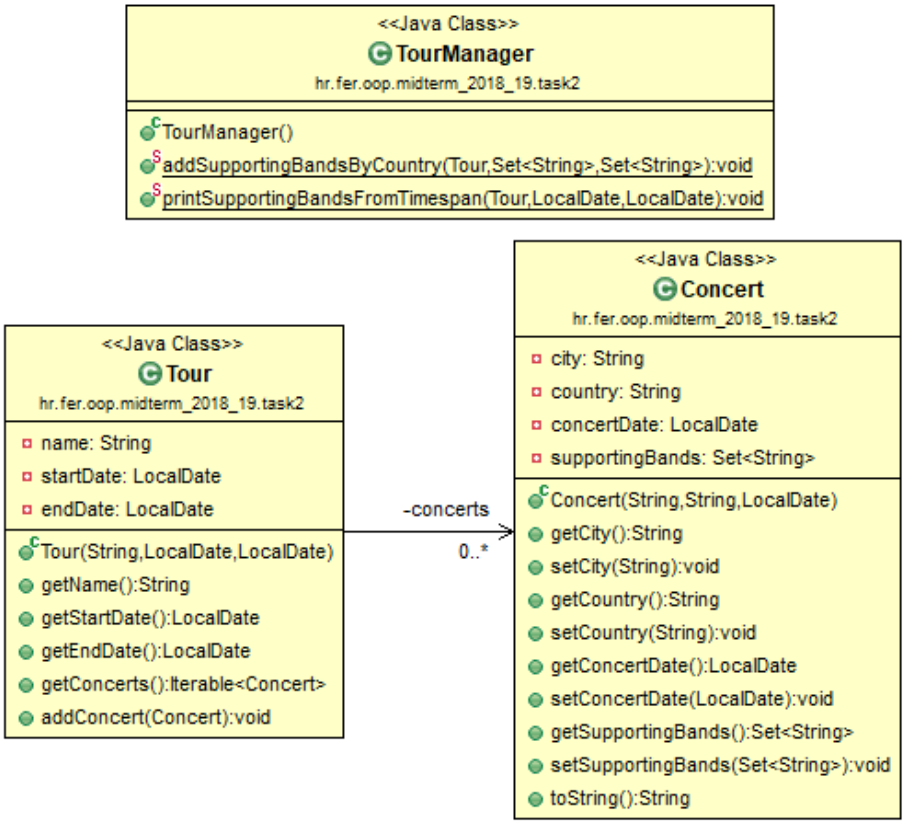
2. ZADATAK (6 bodova)

Klase `Tour` i `Concert` modeliraju glazbene turneje i koncerte u pojedinoj turneji nekog benda. Svaka turneja ima svoj naziv, datum početka i datum završetka. Svaki koncert u turneji sadrži ime grada i ime države u kojoj se koncert održava, datum na koji se koncert održava i popis predgrupa koje sviraju prije glavnog izvođača, pri čemu su imena predgrupa zapisana kao `String`. Postoji menadžer turneje (`TourManager`) koji je odgovoran za upravljanje turnejom i pripadajućim koncertima. U klasi `TourManager` potrebno je napisati sljedeće statičke metode:

a) Metodu `addSupportingBandsByCountry` koja kao argumente prima turneju, popis država i popis predgrupa. Metoda treba u koncerte iz zadane turneje koji se odvijaju u državama iz zadanog popisa država dodati predgrupe iz zadanog popisa predgrupa. Postojeće predgrupe se ne smiju obrisati s koncerta na koji se dodaju nove predgrupe, već se nove predgrupe moraju dodati postojećima. Također, predgrupa se na popisu predgrupa nekog koncerta smije pojavljivati samo jednom. Predgrupe postojećih koncerata inicijalno nisu sortirane, dok se na koncertima koje **metoda modificira** predgrupe **sortiraju abecednim redom**.

b) Metodu `printSupportingBandsFromTimespan` koja kao argumente prima turneju te početni i završni datum. Metoda treba na standardni izlaz ispisati popis predgrupa koje sviraju između zadana dva datuma na turneji (svaku predgrupu je potrebno ispisati u zasebnom retku). Predgrupe se na popisu smiju **pojavljivati samo jednom**, neovisno o tome na koliko koncerata nastupaju. Za uspoređivanje datuma mogu se koristiti metode `boolean isBefore(LocalDate date)` i `boolean isAfter(LocalDate date)` objekta `LocalDate`.

Klase opisane u zadatku, njihove članske varijable i metode mogu se vidjeti na UML dijagramu na slici ispod.



3. ZADATAK (6 bodova)

a) Napišite parametriziranu klasu `BoxOfMemorabilia` koja pohranjuje elemente parametriziranog tipa pri čemu se elementi mogu ponavljati. Klasa mora imati metodu `add` koja prima varijabilni broj elemenata koje treba pohraniti u listu, a kao rezultat ne vraća ništa. Metoda `add` treba ispravno raditi ako se kao jedan od argumenata pošalje `null`: u tom slučaju se takav element ne dodaje u listu te program nastavlja s radom. Također, klasa mora imati metodu `getItems` koja vraća listu pohranjenih elemenata i metodu `getNumberOfItems` koja vraća broj elemenata u listi.
Primjer korištenja klase (programski odsječak ispisuje „`I own 5 memorabilia items`“):

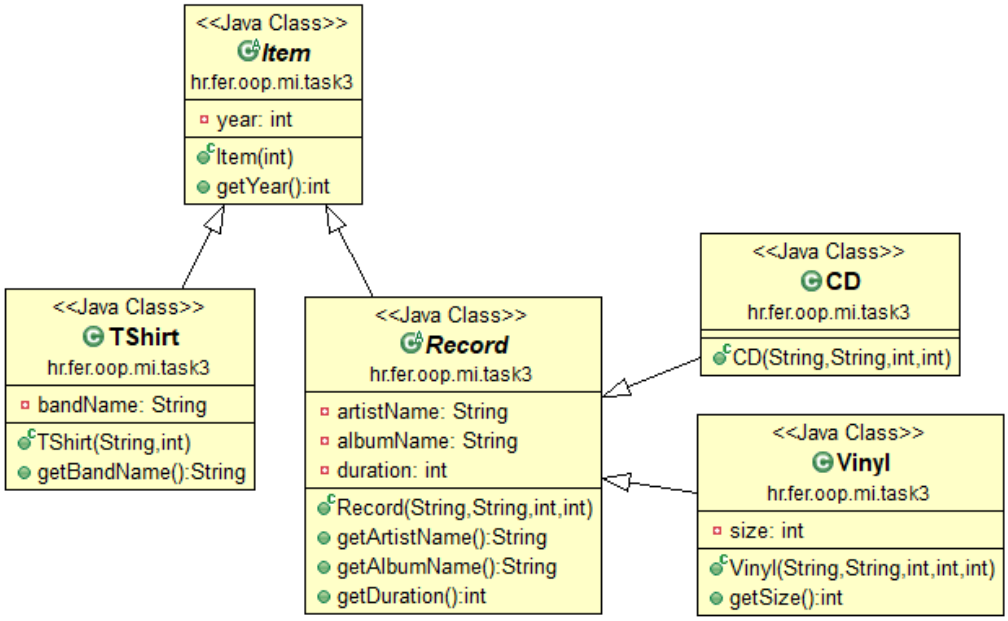
```
CD cd1 = new CD("Foo Fighters", "One by One", 2002, 55);
CD cd2 = new CD("Foo Fighters", "In Your Honor", 2005, 83);
Vinyl vinyl1 = new Vinyl("The Black Keys", "Rubber Factory", 2004, 41, 12);
TShirt tshirt1 = new TShirt("Foo Fighters", 2012);
TShirt tshirt2 = new TShirt("The Black Keys", 2010);

BoxOfMemorabilia<Item> box = new BoxOfMemorabilia<Item>();
box.add(cd1, cd2, vinyl1, tshirt1, tshirt2, null);
List<Item> itemsInBox = box.getItems();
System.out.println("I own " + box.getNumberOfItems() + " memorabilia items.");
```

b) Napišite generičku klasu `BoxOfRecords` koja nasljeđuje klasu BoxOfMemorabilia te je parametrizirana po bilo kojem tipu izvedenom iz klase `Record`. U klasi BoxOfRecords napišite metodu `totalDuration` koja vraća ukupno trajanje svih albuma u kutiji.
Primjer korištenja klase `BoxOfRecords` (programski isječak ispisuje 138, jer je 55 + 83 = 138):

```
BoxOfRecords<CD> boxOfCDs = new BoxOfRecords<>();
boxOfCDs.add(cd1, cd2);
System.out.println(boxOfCDs.totalDuration());
```

Klase opisane u zadatku, njihove članske varijable i metode mogu se vidjeti na UML dijagramu na slici ispod.



4. ZADATAK (5 bodova)

a) Napisati vlastitu iznimku naziva `IllegalMusicalAlphabetException` koja je **provjeravana** pazeći da se zadovolji njena upotreba u b dijelu zadatka. (1 bod)

```
public class IllegalMusicalAlphabetException _____________________________________{




}
```

b) U označeni okvir upišite rezultat izvođenja sljedećeg programa. Sve definirane klase pripadaju istom paketu. (4 boda)

```
public class MusicalAlphabet {
    public static void main(String[] args) {
        String [] scale = new String[] {"c", "e", "55"};
        try {
            for (int i=scale.length; i > 0; i--) {
                try {
                    int x = Integer.parseInt(scale[scale.length - i + 1]);
                    try {
                        System.out.format("%d is number%n", x);
                        throw new IllegalMusicalAlphabetException("Illegal.");
                    }
                    catch (IllegalMusicalAlphabetException exc) {
                           System.out.println(exc.getMessage()); // ispisuje String koji je
                                                                 // bio predan u konstruktoru iznimke
                    }
                    finally {
                        System.out.println("inner finally");
                    }
                }
                catch(IndexOutOfBoundsException exc){
                    System.out.println("Error: 1");
                    throw new IllegalMusicalAlphabetException("Error: 4");
                }
                catch(NumberFormatException exc){
                    System.out.println("Error: 2");
                }
                catch(NullPointerException exc){
                    System.out.println("Error: 3");
                }
                finally {
                    System.out.println("outer finally");
                }
            }
        }
        catch (IllegalMusicalAlphabetException exc) {
            // TO-DO
        }
    }
}
```

Ovdje upisati rješenje 4. b) zadatka