Objektno orijentirano programiranje

Međuispit

20.4.2021.

Ispit nosi ukupno 25 bodova i piše se 150 minuta.
U zadacima nije potrebno pisati dio u kojem se uključuju klase ili paketi klasa (import)
Rješenja je potrebno pisati na stranice ispita. *Ako na nekoj stranici ne budete imali dovoljno mjesta, ostatak možete napisati na zadnju (praznu) stranicu ispita.*

IZJAVA

Tijekom ove provjere znanja neću od drugoga primiti niti drugome pružiti pomoć te se neću koristiti nedopuštenim sredstvima.
Ove su radnje povreda Kodeksa ponašanja te mogu uzrokovati trajno isključenje s Fakulteta.
Zdravstveno stanje dozvoljava mi pisanje ovog ispita.

Vlastoručni potpis studenta: _____

1. zadatak (6 bodova)

Potrebno je dopuniti kod kojim se modeliraju klase koje se koriste u aplikaciji za kontrolu kvalitete i izračun cijene voća u voćnjaku obiteljskog gospodarstva. Trenutno postoje dvije vrste voća: breskva (*Peach*) i trešnja (*Cherry*). Svako voće ima svoj latinski naziv (*species*:*String*), prosječni broj plodova po stablu (*numberOfFruits*:*int*) i trošak proizvodnje po stablu (*productionPrice*:*double*). Breskvu dodatno opisuje prosječni promjer ploda (*peachDiameter*:*int*), dok trešnju karakterizira sadrže li plodovi nametnika tj. crva (*hasWorm*:*boolean*).

Za voće je potrebno definirati dvije javne metode:

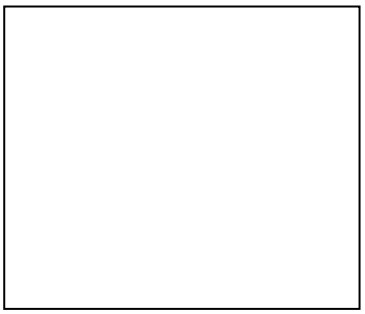
- `double productionPrice(int numberOfTrees)`
- `double myMarketPrice()`

Metoda *productionPrice* vraća trošak proizvodnje za predani broj stabala. Vrijednost se računa množenjem broja stabala i iznosa proizvodnje po stablu te se mora računati uvijek isto bez obzira na vrstu voća.

Metoda *myMarketPrice* računa tržišnu cijenu po plodu. U slučaju breskve tržišna cijena je jednaka proizvodnoj cijeni uvećanoj za 40% ako je promjer breskve u rasponu [2,8], a inače za 30%. U slučaju trešnje, tržišna cijena je jednaka proizvodnoj cijeni uvećanoj za 20% ili 50% u ovisnosti ima li trešnja crva ili ne.

Napomena: Odgovarajući *getteri* i *setteri* su već napisani u svakoj klasi (ne treba ih pisati), ali su izostavljeni zbog preglednosti.

```
public _____ {
	private String species;
	private int numberOfFruits;
	private double productionPrice;

	public Fruit(String species, int numberOfFruits, double productionPrice) {
		this.species = species;
		this.numberOfFruits = numberOfFruits;
		this.productionPrice = productionPrice;
	}









}
```



```
public _____ {
	private boolean hasWorm;
	public Cherry(String species, int numberOfFruits,
			double productionPrice, boolean hasWorm) {



	}









}
```

```
public _____ {
	private int peachDiameter;
	public Peach(String species, int numberOfFruits,
			double productionPrice, int peachDiameter) {



	}









}
```

2. zadatak (7 bodova)

Potrebno je modelirati sustav multimedijske knjižnice koji je opisan u nastavku teksta. **Vaš zadatak je nadopuniti predložak UML dijagrama klasa ovog sustava koji se nalazi na sljedećoj stranici**. Pritom **možete dodati nove klase** (ili sučelja) ako smatrate da je to potrebno**. Nije potrebno pisati programski kod sustava i to neće donositi nikakve bodove**. **Sve članske varijable se inicijaliziraju u konstruktorima osim ako je eksplicitno navedeno drugačije.**

Na ovom predlošku pažljivo navedite:

- oznake za sučelje (**I**), enumeraciju (**E**), apstraktnu (**A**) ili običnu klasu (**C**)
- oznake za apstraktnu metodu (**A**), oznake za konstruktore (**C**) te oznake za statičku (**S**) i finalnu (**F**) metodu ili atribut
- modifikatore vidljivosti metoda i atributa (**+** public, **#** protected, **-** private, **~** package-private)
- tipove atributa, povratnih vrijednosti i argumenata
    - nazivi argumenata se ne navode, već samo njihovi tipovi
- odgovarajuće strelice da naznačite nasljeđivanje klasa (puna linija ⟶▷), implementaciju sučelja (iscrtkana linija ⋯▷) ili asocijaciju (puna linija → s navedenim nazivom varijable, modifikatorom vidljivosti i odgovarajućom kardinalnosti)

Multimedijska knjižnica (`MultimediaLibrary`) služi za upravljanje skupom multimedijskih stavki (`MultimediaItem`) sadržanih u polju, gdje je maksimalna veličina polja zadana kao argument konstruktora. Multimedijskim stavkama se upravlja metodama dostupnim u knjižnici. Stavke se mogu dodati i ukloniti iz knjižnice pomoću metoda `addItem` i `removeItem`. Metode za dodavanje i uklanjanje ne vraćaju ništa. Broj stavki se može dobiti pozivom metode `count`, a element na pojedinoj poziciji metodom `itemAt`.

Svaka multimedijska stavka (`MultimediaItem`) ima svoj naslov (`title`) tipa `String`. Naslov se može dohvatiti i postaviti (`getTitle`, `setTitle`). Svaka stavka se predstavlja određenom ikonom za što je predviđena metoda *getIcon* (povratna vrijednost je polje bajtova).

Postoje četiri vrste multimedijskih stavki: video (`Video`), audio (`Audio`), fotografija (`Photo`) i popis za reprodukciju (`Playlist`).

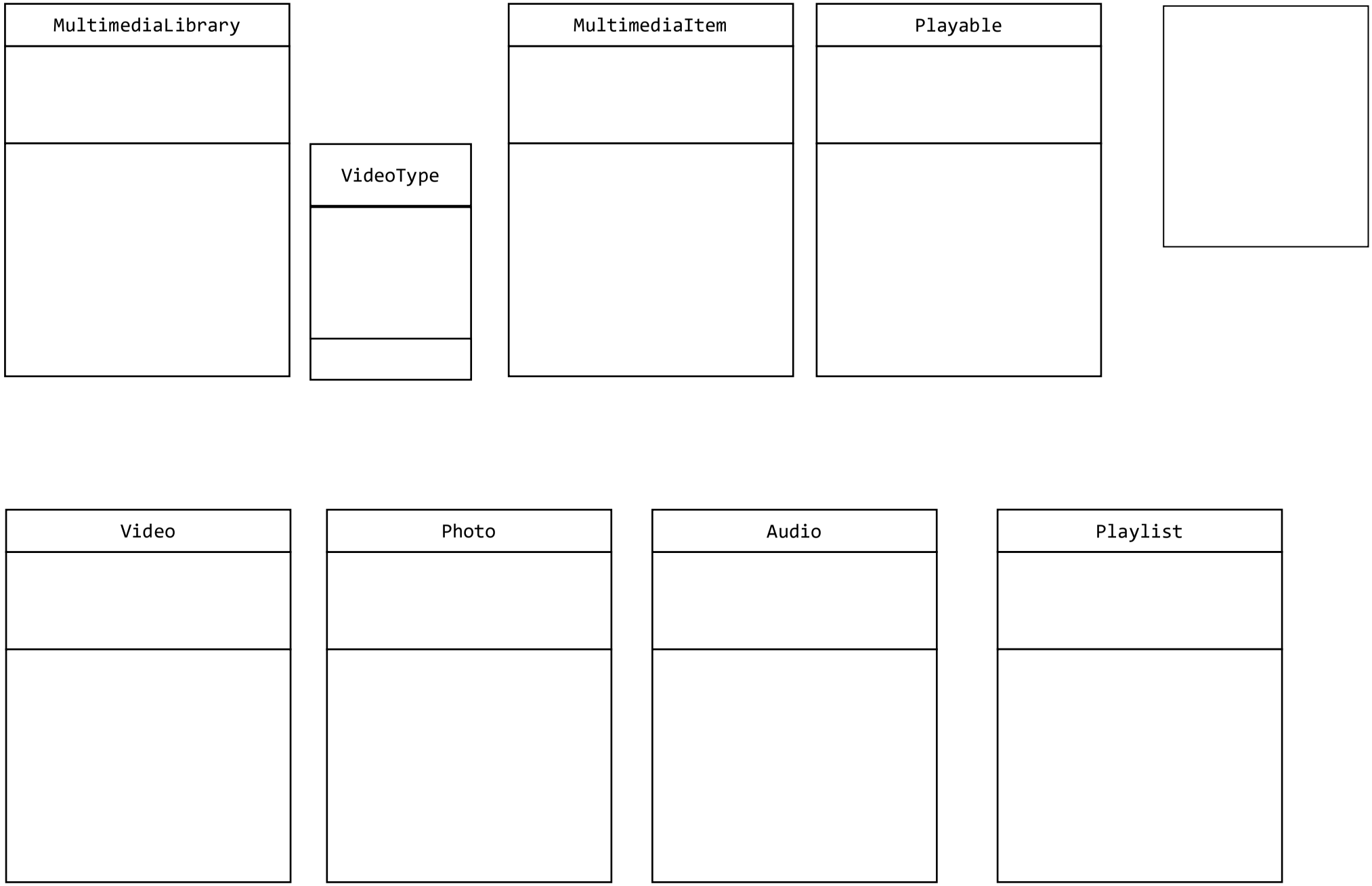
Video opisuju varijabla `type` koja može biti u jednom od 3 stanja tipa `VideoType` (`MOVIE`, `TV_EPISODE` i `RECORDING`) i varijabla `duration` tipa `double` koja opisuje trajanje videa. Navedene varijable se postavljaju u konstruktoru i ne mogu se mijenjati, a dohvat se vrši *getterima* (`getType`, `getDuration`).

Audio opisuju tri varijable: `artist` tipa `String`, `album` tipa `String` i `duration` tipa `double`. Artist i album se mogu postaviti i dohvatiti putem odgovarajućih *gettera* i *settera* (`getArtist`, `setArtist`, `getAlbum`, `setAlbum`) dok se `duration` postavlja isključivo putem konstruktora i ne može se promijeniti, a dohvatiti se može putem metode `getDuration`.

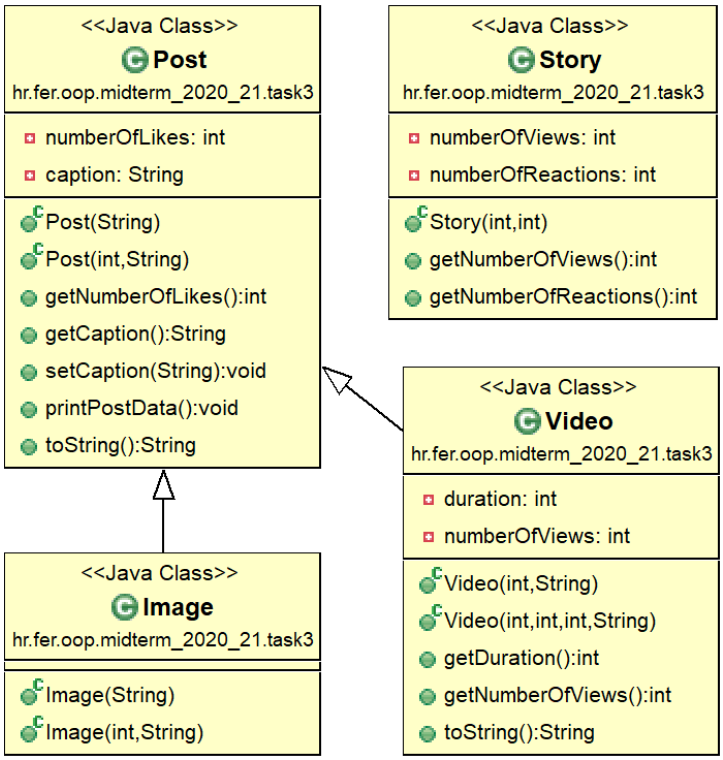
Video i Audio se mogu reproducirati (`Playable`). Objekt se može reproducirati ako implementira metode `void play()` i `void pause()`.

Fotografije su opisane pomoću tri varijable: `height` tipa `int`, `width` tipa `int` i `dateCreated` tipa `String`. Visina i širina su argumenti konstruktora i nepromjenjivi, a dateCreated se kreira automatski ovisno o trenutku stvaranja objekta. Vrijednosti se mogu dohvatiti pomoću odgovarajućih *gettera* (`getHeight`, `getWidth`, `getDateCreated`). Fotografija se može prikazati metodom `display()`;

U svakom popisu za reprodukciju se nalazi polje `playables` koje sadrži elemente koji se mogu reproducirati. Maksimalna veličina popisa zadaje se konstruktorom, a inicijalno može sadržavati varijabilni broj elemenata. Popis za reprodukciju nudi metode `add` i `remove` koje primaju element koji je potrebno dodati ili ukloniti te metode za skok na sljedeći ili prethodni element `next()` i `previous()`. Navedene metode ne vraćaju ništa. Popis za reprodukciju se, kao video i audio, također može reproducirati.


MultimediaLibrary
VideoType
MultimediaItem
Playable
Video
Photo
Audio
Playlist

3. zadatak (6 bodova)



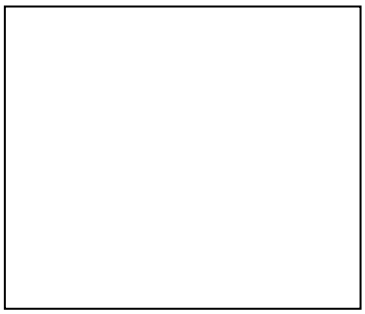
Klase *Post*, *Story*, *Video* i *Image* zadane su kao na slici s desne stane. Vaš zadatak je

a) Napisati parametriziranu klasu *WebEntry* koja sadrži članske varijable *author* i *date* (oboje tipa *String*) te parametriziranu člansku varijablu *entry*, a zatim napisati konstruktor kojim se postavljaju vrijednosti članskih varijabli.
Nije potrebno pisati gettere i settere (ali u b i c dijelu zadatka možete pretpostaviti da su napisani).

b) Napisati klasu *InstagramStory* koja predstavlja *WebEntry* tipa *Story*.
Ova klasa treba implementirati metodu *calculateStoryImpact* koja kao argument prima ukupan broj pratitelja nekog korisnika. Metoda vraća utjecaj (tipa *double*) koji se računa kao zbroj pregleda (*numberOfViews*) i reakcija (*numberOfReactions*) podijeljen s brojem pratitelja.

c) Napisati klasu *InstagramPost* koja predstavlja *WebEntry* ograničen na *Post* i tipove izvedene iz njega te implementira metodu *calculatePostImpact* koja kao argument prima ukupan broj lajkova na profilu nekog korisnika. Metoda vraća utjecaj sadržaja (tipa *double*) koji se računa kao broj lajkova tog sadržaja (*numberOfLikes*) podijeljen s ukupnim brojem lajkova.

```
// WebEntry.java
```



```
// InstagramStory.java
```

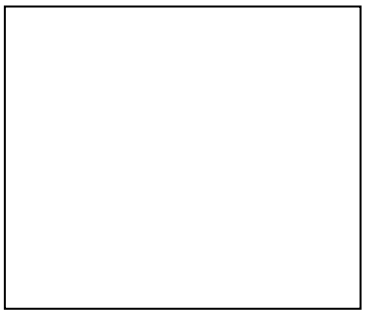
```
// InstagramPost.java
```

4. zadatak (6 bodova)

U označeni okvir upišite što ispisuje sljedeći program. Ako program završava neuhvaćenom iznimkom, na dnu ispisa napisati *Exception in main* i navesti vrstu iznimke.
Sve definirane klase pripadaju istom paketu i u svakoj klasi na vrhu su uključeni potrebni paketi te `import static java.lang.System.out;`

```
public class Main {
  public static void main(String[] args)
                        throws Exception {
    Game g1 = null;
    Game g2 = null;
    VideoGame vg1 = null;
    VideoGame vg2 = null;

    try (Game g3 = new Game(10, "NFS")) {
      g1 = new VideoGame(100, "COD");
      vg1 = new VideoGame(0, "Cyberpunk 2077");
      vg2 = new VideoGame(3, "Minecraft");
      g2 = new Game(20, "Risk");

    }
    catch (NoCopiesException e) {
      out.println(e.getMessage());
    }
    catch (Exception e) {
      out.println("Game sales exception");
    }
    finally {
      out.println("Setup done");
      vg1 = new VideoGame(50, "Grim Dawn");
    }

    try {
      vg1.buyGame(5);
      vg2.buyGame(60);
      g1.buyGame(7);
    }
    catch (NullPointerException e) {
      out.println("Null pointer exception");
    }
    catch (Exception e) {
      out.println("Exception");
    }
    finally {
      out.println("Sale finished!");
      ((Game) vg1).close();
    }
    g1.buyGame(7);
    out.println(g1.numberOfCopies);
  }
}
```

OVDJE UPISATI RJEŠENJE:



```
public class NoCopiesException extends Exception {
     public NoCopiesException(String msg) {
          super(msg);
     }
}
```

```
public class Game implements Closeable {
  String name;
  int numberOfCopies;
  public Game(int numberOfCopies, String name) throws NoCopiesException {
      this.name = name;
      System.out.println(name);
      if (numberOfCopies <= 0)
          throw new NoCopiesException("Number of copies must be at least 1!");
      else
          this.numberOfCopies = numberOfCopies;
  }
  public void buyGame(int num) {
      if (num > numberOfCopies)
          throw new IllegalArgumentException("Not enough copies!");
      else {
          System.out.println("Number of copies purchased: " + num);
          numberOfCopies = numberOfCopies - num;
      }
  }
  public int getNumOfCopies() {
      return numberOfCopies;
  }
  @Override
  public void close() throws IOException {
      System.out.println(name + " removed");
  }
}
```

```
public class VideoGame extends Game {
  int sold;
  public VideoGame(int numOfCopies, String name) throws NoCopiesException {
      super(numOfCopies, name);
  }
  @Override
  public void buyGame(int num) {
      super.buyGame(num);
      sold += num;
      System.out.println("Sold " + num + " copies.");
  }
  @Override
  public void close() throws IOException {
      System.out.println("Video game " + name + " deleted.");
      super.close();
  }
}
```