

Započeto Srijeda, 6 Lipanj 2018, 16:41

Stanje završen

Završeno Srijeda, 6 Lipanj 2018, 16:56

Proteklo vrijeme 15 min 1 sek

Bodovi 13,18/30,00

Ocjena **4,39** od maksimalno 10,00 (**44%**)

Pitanje 1

Netočno

Broj bodova:
0,00 od 1,00

Samo unutarnje klase mogu biti određene rezerviranom riječju **static**.

Odaberite jedan odgovor:

- ☐ Točno
- ☒ Netočno ❌

Pitanje 2

Nije
odgovoreno

Broj bodova od
1,00

Drugačiji mehanizam od preklapanja (eng. override) jeste mehanizam
(eng. overloading) metoda.

istih ❌

Pitanje 3

Točno

Broj bodova:
1,00 od 1,00

Zamislamo da imamo klasu Device i unutar nje **statičku** metodu **someMethod()**

→ preciznije:

```
public static void someMethod() {  
    // commands  
}
```

U tom slučaju metodi:

Odaberite jedan odgovor:

- ☐ a. Možemo pristupiti bilo kojim objektom koji je dobiven iz podklase polazne klase Device
- ☒ b. Možemo pristupiti bez kreiranja objekata klase Device ✔️
- ☐ c. Ne možemo metodu definirati kao static
- ☐ d. Možemo pristupiti samo unutar klase Device
- ☐ e. Možemo pristupiti tek nakon kreiranja nekog objekta klase Device

Vaš odgovor je točan.



Pitanje 4

Točno

Broj bodova:
1,00 od 1,00

poruke zna definicije objekte razumije atribut metode objašnjava klase prevodi

Na predavanjima i vježbama smo nekoliko puta ukazali na činjenicu kako nije dobra praksa da korisnicima prikazujemo generičke poruke ✓ iznimki/grešaka (npr. e.printStackTrace()), već je poželjno generirati prilagođenu poruku koju korisnik razumije ✓ i koja mu naznačava u čemu je problem te što treba poduzeti.

Vaš odgovor je točan.

Pitanje 5

Točno

Broj bodova:
1,00 od 1,00

Imamo primjer u kojem smo namjerno postavili da korisnički unos s tipkovnice čitamo kao Integer. Što će se u slučaju unosa broja 38 dobiti kao konzolni izlaz? Pripadni kod je:

```
import java.util.Scanner;

public class App2 {

    public static void main(String[] args) {

        Object occupation;

        Scanner sc = new Scanner(System.in);

        System.out.println("Please enter your occupation: ");
        try {
            occupation = sc.nextInt();

            if (!(occupation instanceof String)) {

                throw new Exception("Input is not a string!");

            } else {
                System.out.println(occupation);
            }

        }

        catch (Exception e) {
            System.out.println(e.getMessage());
        }

    }

}
```

Odaberite jedan odgovor:

- ☐ a. Konzolni izlaz je upravo uneseni broj
- ☐ b. printStackTrace()
- ☐ c. null
- ☐ d. Konzolni izlaz ne prikazuje ništa
- ☒ e. "Input is not a string!" ✓

Vaš odgovor je točan.



Pitanje **6**

Djelomično
točno

Broj bodova:
0,33 od 1,00

Što su ispravni primjeri overloadinga metode **public void openAccess()** (zamislite da metoda ima pripadajuće tijelo / implementaciju):

Odaberite jedan ili više odgovora:

- ☐ a. public void accessOpen()
- ☐ b. public ArrayList <String> openAccess(long ups)
- ☒ c. public void openAccess(int cnt, String con) ✓
- ☐ d. public int openAccess(String connection)
- ☐ e. public String openAccess()

Vaš odgovor je djelomično točan.

Broj točnih odgovora: 1

Pitanje **7**

Točno

Broj bodova:
1,00 od 1,00

Zamislimo primjer u kojem nakon try bloka slijedi sljedeći catch blok:

```
catch (FileNotFoundException ex) {  
    ex.printStackTrace();  
}
```

Objašnjenje catch bloka je:

Odaberite jedan odgovor:

- ☐ a. Generira se objekt klase FileNotFoundException i on poziva metodu printStackTrace() kojom se ispisuje samo linija koda u kojoj je greška nastala
- ☐ b. Generira se klasa s metodom printStackTrace()
- ☐ c. Ispisuje se korisnički definirana poruka o izuzetku
- ☐ d. Ovako definiran catch blok je neispravno definiran i nema nikakvo značenje
- ☒ e. Generira se objekt klase FileNotFoundException i on poziva metodu printStackTrace() kojom se ispisuje trag o grešci u cijelom stogu (Stack) ✓

Vaš odgovor je točan.

Pitanje **8**

Točno

Broj bodova:
1,00 od 1,00

Runtime exception uvijek **mora** biti obuhvaćena mehanizmom upravljanja iznimkama.

Odaberite jedan odgovor:

- ☐ Točno
- ☒ Netočno ✓



Pitanje **9**

Netočno

Broj bodova:
-0,50 od 1,00

Često smo se pri rješavanju primjera na predavanjima i vježbama susreli s konstrukcijom oblika:

```
try(FileInputStream fis = new FileInputStream("fileName.txt")){
```

```
//code
```

```
}
```

Za koji smo naglasili da automatski zatvara korišteni resurs (nije potreban eksplicitni `fis.close()`). Takav mehanizam nazivamo:

Odaberite jedan odgovor:

- ☒ a. finally ❌
- ☐ b. finalized
- ☒ c. try-catch
- ☐ d. try-with-resources
- ☐ e. polimorfizam

Vaš odgovor nije točan.

Pitanje **10**

Točno

Broj bodova:
1,00 od 1,00

Jedan od preporučenih mehanizama za postavljanje vrijednosti atributa objekata i njihov dohvat jest mehanizam setters/getters ▼ ✓ .

Pitanje **11**

Točno

Broj bodova:
1,00 od 1,00

specifikatori modifikatori konstruktori finalizatori generici

Mehanizam koji je riješio problem greški u izvođenju koda (tzv. runtime errors) kod struktura podataka direktnim - eksplicitnim navođenjem tipa objekata (i ključeva tamo gdje je potrebno) nazivamo generici ✓ .

Vaš odgovor je točan.



Pitanje 12

Točno

Broj bodova:
1,00 od 1,00

Mehanizam upravljanja iznimkama se u literaturi na eng. naziva:

Odaberite jedan odgovor:

- ☐ a. Creating Exceptions Code
- ☐ b. Error Handling
- ☐ c. Using Exceptions
- ☒ d. Exceptions Handling ✓
- ☐ e. Exceptions Management

Vaš odgovor je točan.

Pitanje 13

Točno

Broj bodova:
1,00 od 1,00

Kada je neko polje (atribut) klase označen s rezerviranom riječju **static** onda je to polje zajedničko za sve objekte ✓ te klase. Sjetite se primjera s vježbi kada smo za counter stavljali static kako bismo kreirali automatski inkrement za id kupaca.

Pitanje 14

Netočno

Broj bodova:
0,00 od 1,00

Tipičan primjer polimorfizma i to dinamičkog tipa jeste preklapanje (eng. override) metode toString koja metoda osnovne klase Object:

`String toString()` Returns a string representation of the object.

U jednom našem primjeru s vježbi implementirali smo je za klasu Robot na sljedeći način:

```
@Override
public String toString() {
    return "[Robot id = " + id + ", description = " + description + "];"
}
```

Ovaj mehanizam nam omogućava da koristimo isto deklarirane metode objekte ✗, a dobijemo drugačije ponašanje / rezultate. Drugim riječima, u trenutku pizvođenja znat će se što ta metoda radi kod objekta klase Robot, a što kod objekta klase Person.

Pitanje 15

Točno

Broj bodova:
1,00 od 1,00

Jedan try blok može imati više catch blokova.

Odaberite jedan odgovor:

- ☒ Točno ✓
- ☐ Netočno

Pitanje 16

Točno

Broj bodova:
1,00 od 1,00

Za klasu koju ne želimo proširivati koristit ćemo rezerviranu riječ final ✓.



Pitanje 17

Netočno

Broj bodova:
0,00 od 1,00

Svi izuzeci koji se provjere za vrijeme kompilacije nazivaju se (eng.) ✗ exceptions, a za vrijeme samog izvršavanja provjeravaju se tzv. (eng.) ✗ exceptions.

Pitanje 18

Netočno

Broj bodova:
0,00 od 1,00

JAVA i drugi OOP imaju mehanizam dinamičkog alociranja memorije za objekte u vidu spremnika koje nazivamo ✗.

Pitanje 19

Točno

Broj bodova:
1,00 od 1,00

Static metode nemaju pristupa nestatičnim atributima klase.

Odaberite jedan odgovor:

- ☒ Točno ✓
- ☐ Netočno

Pitanje 20

Točno

Broj bodova:
1,00 od 1,00

Kada želimo da klasa ima pristup unutar ✓ paketa nećemo koristiti nikakav specifikator pristupa.

Pitanje 21

Netočno

Broj bodova:
-0,40 od 1,00

Modifikator ili kako ga često nazivamo i specifikator pristupa **private** nekog atributa klase određuje:

Odaberite jedan odgovor:

- ☐ a. Pristup od strane svih klasa
- ☒ b. Pristup unutar istog paketa ✗
- ☐ c. Samo podklase imaju pristup tom atributu
- ☐ d. neograničeni pristup
- ☒ e. Pristup unutar klase

Vaš odgovor nije točan.

Pitanje 22

Netočno

Broj bodova:
0,00 od 1,00



List, Set i Queue predstavljaju tzv. Java ✗ (2 riječi).



Pitanje **23**

Netočno

Broj bodova:
0,00 od 1,00

U sučeljima je ponekad korisno navoditi i metode koje imaju direktnu realizaciju određenu tijelom metode, ali tada metoda mora biti označena kao `void`   .

~~PUBLIC~~

Pitanje **24**

Netočno

Broj bodova:
-0,25 od 1,00

Pretpostavimo da imamo dvije strukture podataka:

```
HashSet<Person> set1;
```

```
TreeSet<Person> set2;
```

Ukoliko želimo imati samo jednu metodu koja će ispisati sve članove skupova set1 i set2 onda bi ona morala biti definirana kao:

Odaberite jedan odgovor:


- ☐ a.

```
public void printAllElements(TreeSet<Person> set){  
    //code  
}
```
- ☒ b.

```
public void printAllElements(Set<Person> set){  
    //code  
}
```
- ☐ c.

```
public void printAllElements(HashSet set) {  
    //code  
}
```
- ☐ d.

```
public void printAllElements(Set set1){  
    // code  
}
```



Vaš odgovor nije točan.



Pitanje 25

Točno

Broj bodova:
1,00 od 1,00

Pretpostavimo da imamo ispravno napisan kod i njegov dio s hvatanjem iznimki je:

```
} catch (Exception ex ) {  
    System.out.println(ex.getMessage());  
}  
} catch (InputMismatchException ine) {  
    System.out.println("Invalid input!");  
}  
}
```

Nedostatak ovako formuliranih blokova s hvatanjem iznimki je:

```
} catch (InputMismatchException e) {  
    sc.next(); // clear leftover garbage in input buffer  
    System.out.println(" Invalid input \t -> Please enter a integer value !");  
}  
} catch (Exception ex) {  
    System.out.println("Error: " + ex.getMessage());  
}  
}
```

Odaberite jedan odgovor:

- ☐ a. Uvijek će se izvršiti samo drugi catch blok pri pojavi iznimki
- ☐ b. Nema nedostataka - sve je u redu formulirano
- ☒ c. Uvijek će se izvršiti samo prvi catch blok pri pojavi iznimki ✓
- ☐ d. InputMismatchException je nepostojeća klasa, pa ćemo imati grešku kompajliranja
- ☐ e. Nikad se ne može izvršiti niti jedan catch blok pri pojavi iznimki

Vaš odgovor je točan.

Pitanje 26

Nije
odgovorenoBroj bodova od
1,00

ključevi



mapa moraju biti jedinstveni, a vrijednosti / objekti se mogu

ponavljati



Pitanje 27

Nije
odgovorenoBroj bodova od
1,00

Mehanizam upravljanja iznimkama je koncept isključivo vezan uz OOP.

Odaberite jedan odgovor:

- ☐ Točno
- ☒ Netočno



Pitanje 28

Nije
odgovorenoBroj bodova od
1,00

Kod upravljanja iznimkama koristili smo **try-catch** blokove, ali smo spomenuli i blok koji nije obavezan, ali se može navesti uz ta dva bloka i onda će uvijek biti izvršen (osim u nekim ekstremnim slučajevima). Taj blok smo nazvali:

Odaberite jedan odgovor:

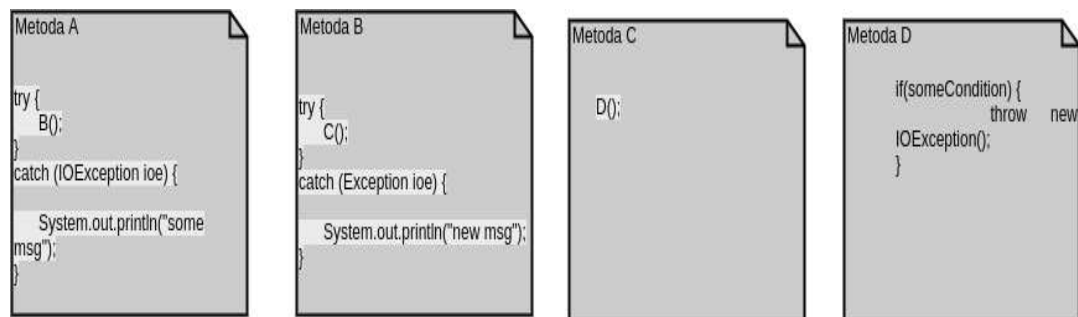
- ☐ a. final block
- ☐ b. static blok
- ☐ c. important blok
- ☐ d. while blok
- ☒ e. finally blok

Vaš odgovor nije točan.

Pitanje 29

Nije
odgovorenoBroj bodova od
1,00

Rezultat pozivanja niza metoda kako je prikazano na slici:



U slučaju istinitosti uvjeta someCondition dat će sljedeći rezultat:

Odaberite jedan odgovor:

- ☐ a. Neće se izvršiti niti jedna metoda
- ☒ b. ispisat će se poruka "new msg"
- ☐ c. ioe.printStackTrace()
- ☐ d. Ispisat će se poruka "some msg"

Vaš odgovor nije točan.




Pitanje **30**

Nije
odgovoreno

Broj bodova od
1,00

prekidima Throwable greškama List nelogičnostima objektima izuzecima Error
problemima Object

Na predavanjima i vježbama smo obradili izuzetke (eng. Exceptions). Pronađite java api za Exception i pogledajte njezinu superklasu. Ta klasa  **[Throwable]** je superklasa svim

 **[greškama]** i  **[izuzecima]** u JAVI.

Vaš odgovor nije točan.

