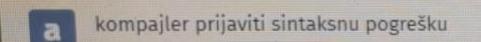
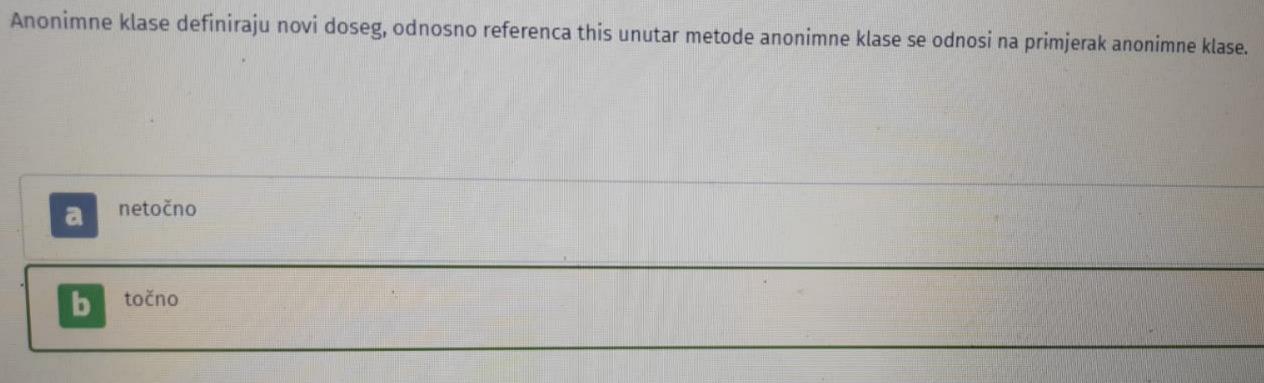
Ako vlastita klasa Student ne implementira nijedno sučelje niti je eksplicitno izvedena iz neke druge klase te u programu postoji sljedeći odsječak koda

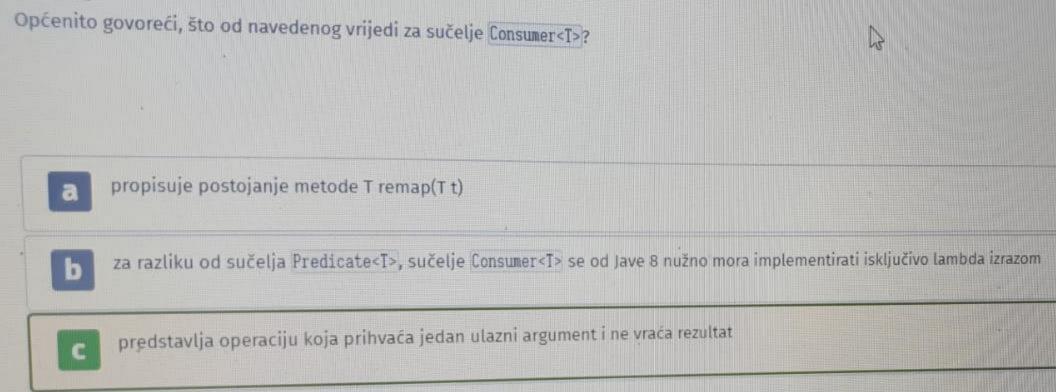
```
Student s = new Student("Horvat", "Hrvoje", "19923492");
Set<Student> students = new TreeSet<>();
students.add(s);
```

tada će

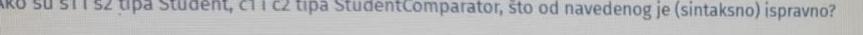


- program vršiti usporedbu studenata na temelju metode compareTo iz klase Object
- se prilikom izvođenja programa dogoditi iznimka

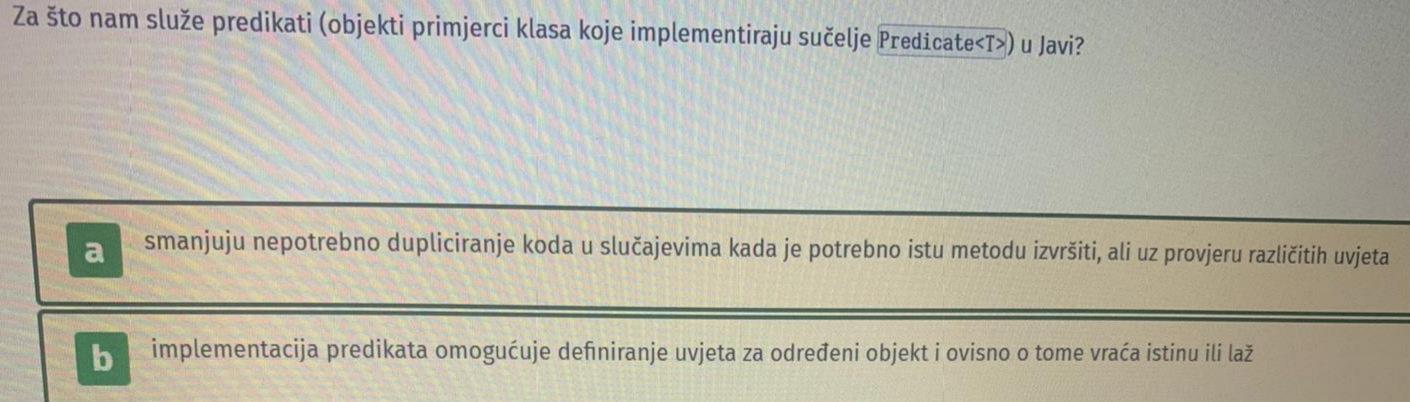




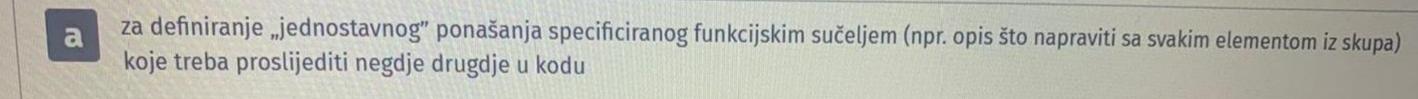
Neka klasa Student ne implementira nijedno sučelje niti je eksplicitno izvedena iz neke druge klase te neka je definirana klasa StudentComparator takva da implementira Comparator<Student>. Ako su s1 i s2 tipa Student, c1 i c2 tipa StudentComparator, što od navedenog je (sintaksno) ispravno?



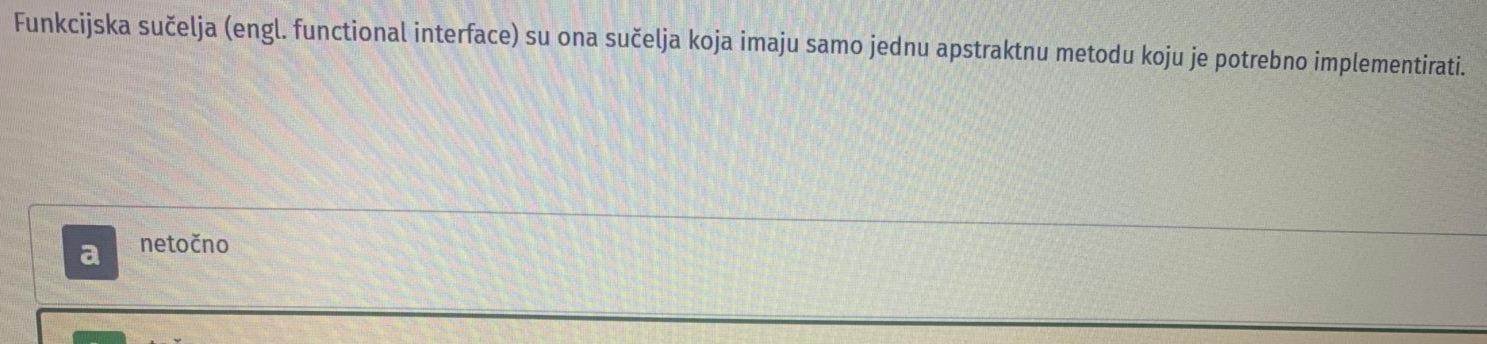




Kada je pogodno koristiti lokalnu klasu?



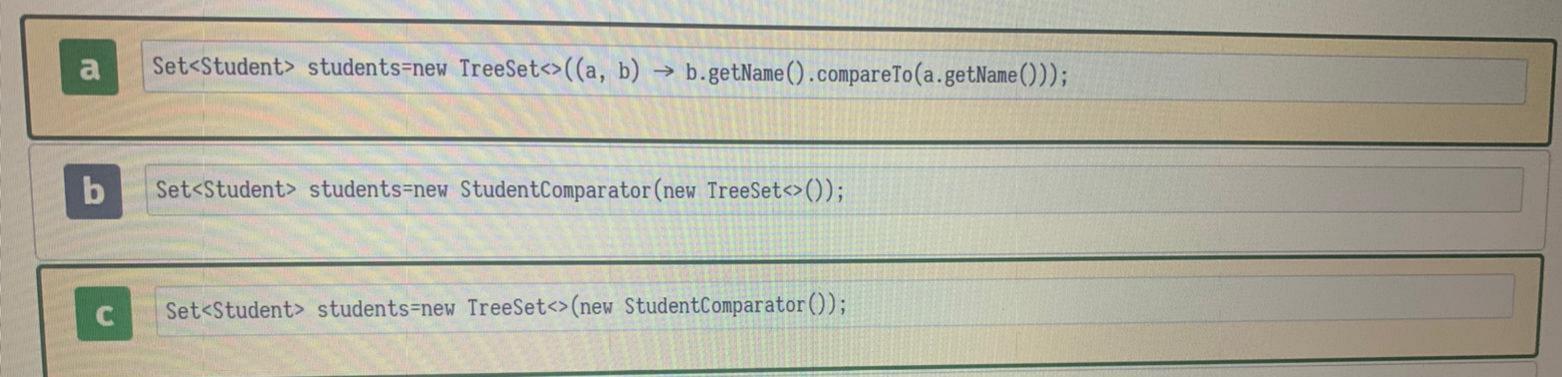
kad se klasa ne koristi nigdje van metode u kojoj je definirana, ali postoji više instanci klase, kad trebamo konstruktor ili je jednostavno potreban imenovani tip zbog dodatnih metoda ili varijabli



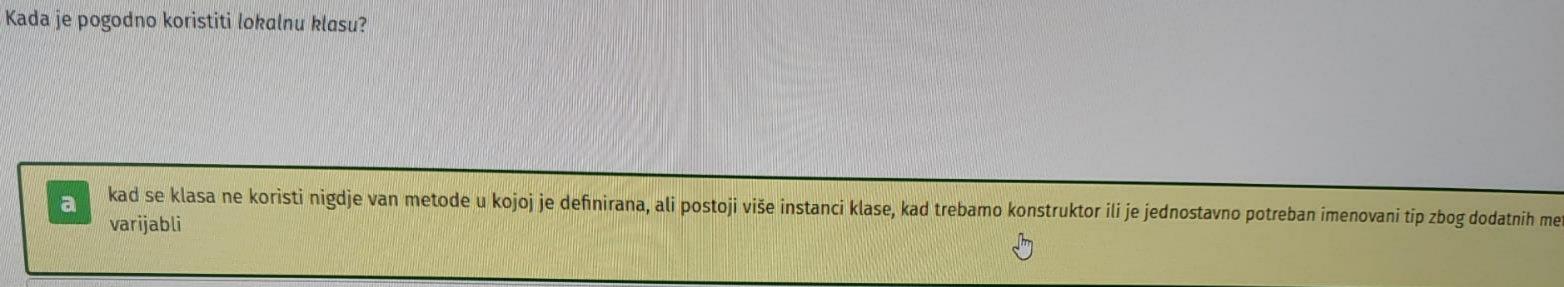
točno

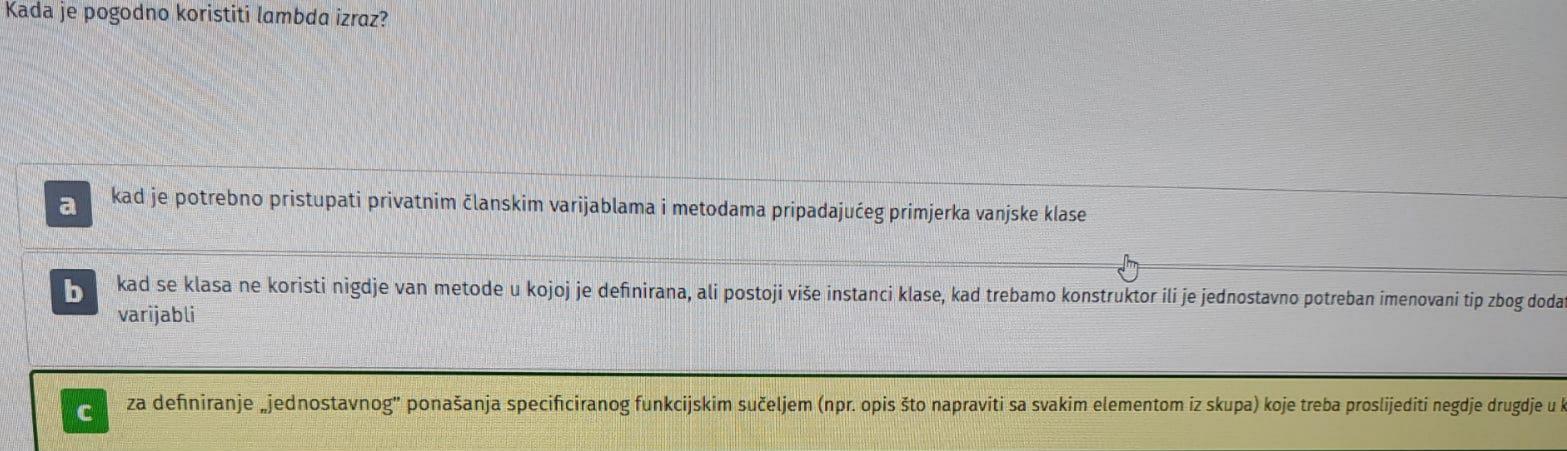
Neka je početak definiracije klase Student public class Student implements Comparable < Student > te neka su s1 i s2 tipa StudentŠto je od navedenog (sintaksno) ispravno? Comparable.compareTo(s1, s2) Student.compareTo(s1, s2) s2.compareTo(s1) Comparable < Student > . compare To(s1, s2) s1.compareTo(s1)

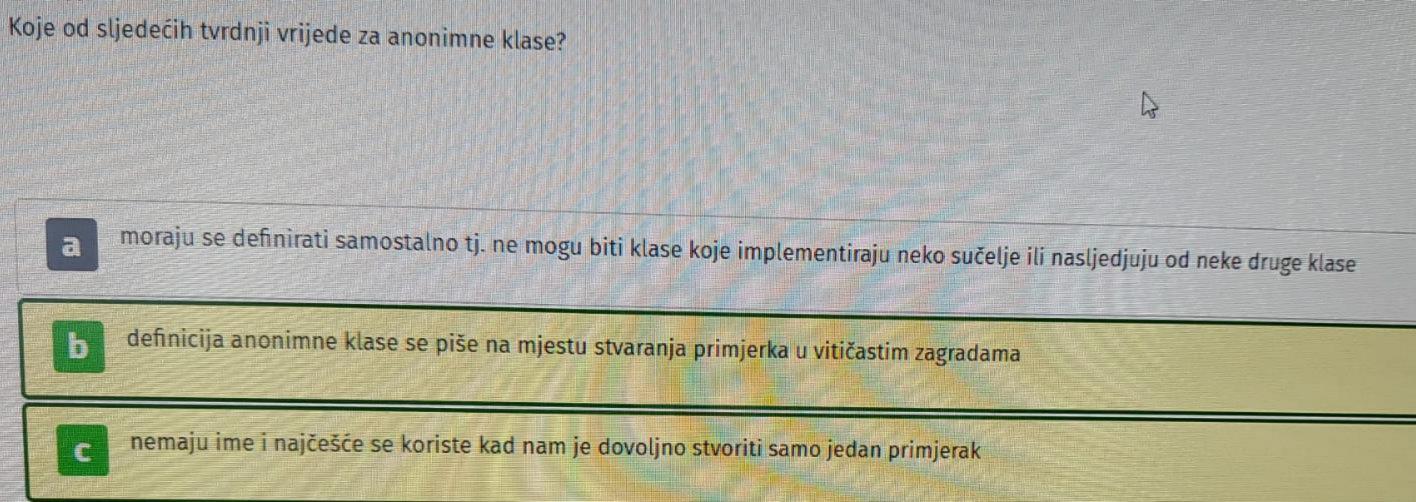
Neka klasa Student ne implementira nijedno sučelje niti je eksplicitno izvedena iz neke druge klase te neka je definirana klasa StudentComparator takva da implementira Comparator<Student> Uz pretpostavku da klasa Student ima metode getName (vraća String) i getGrade(vraća int) što od navedenog je (sintaksno) ispravno?



```
Što ispisuje program?
 class Student {
     String id, name;
     public Student(String i, String n) {
          id = i:
          name = n:
 3
 public class Main [
      public static void main(String[] args) {
           Student perol = new Student("123456", "Pero");
           Student pero2 = new Student("123456", "Pero");
           System.out.print(perol.equals(pero2) + ", ");
           System.out.println(perol.equals(perol));
      3
  3
            true, false
            true, true
            false, false
             Program je sintaksno ispravan, ali baca iznimku
             false, true
```







Koje su tipične naredbe za početak metode equals koja je nadjačana u klasi Student?

```
MOverride
public boolean equals(Object obj) {
    if (obj = null) return false;
    if (!(obj instanceof Student))
       return false;
```

Neka je početak definiracije klase Student public class Student implements Comparable<Student> te neka su s1 i s2 tipa StudentŠto je od navedenog (sintaksno) ispravno? sl.compareTo(s1) Comparable.compareTo(s1, s2) s2.compareTo(s1)

Prirodni	poredak, odnosno prirodni komparator za neku klasu T definira se
a	definiranjem nekog komparatora koji se svodi na usporedbu članskih varijabli
Ь	implementacijom sučelja Comparable i metode compareTo(T first, T second)
С	nadjačavanjem metode compareTo iz klase Object
d	implementacijom sučelja Comparable i metode compareTo(T other)

Neka je p	početak definiracije klase Student
pub	lic class Student implements Comparable <student></student>
te neka s	su s1 i s2 tipa StudentŠto je od navedenog (sintaksno) ispravno?
a	Comparable.compareTo(s1, s2)
b	Student.compareTo(s1, s2)
C	s2.compareTo(s1)
d	Comparable <student>.compareTo(s1, s2)</student>
е	s1.compareTo(s1)

oji od n	avedenih potpisa su potpisi metoda equals i hashCode iz klase java.lang.Object
a	int equals(Object obj)
Ь	<pre>int hashCode(Object obj)</pre>
С	boolean equals(Object obj1, Object obj2)
d	int hashCode()
е	boolean equals(Object obj)

Sučelje I	terable definira metodu za kreiranje iteratora. Koja tvrdnja je ispravna:
a	iterator pamti svoju poziciju neovisno o drugim stvorenim iteratorima
Ь	djelovanje iteratora ovisi o drugim stvorenim iteratorima iz iste kolekcije
С	svaka kolekcija implementira Iterable koji može stvoriti takav iterator

Kada je pogodno koristiti anonimnu klasu?	
a	kad je potrebno pristupati privatnim članskim varijablama i metodama pripadajućeg primjerka vanjske klase
b	ad se klasa ne koristi nigdje van metode u kojoj je definirana, ali postoji više instanci klase, kad trebamo konstruktor ili je jednostavno potreban imenovani tip zbog dodatnih metoda ili varijabli
С	kad je potrebno koristiti varijable iz koda koji se nalazi izvan anonimne klase
d	Kad je potrebno napisati dva ili više konstruktora
е	u slučajevima kad lambda izraz nije prikladan, kad je potrebno kreirati samo jedan primjerak objekta iz klase u kojoj se mogu definirati dodatni atributi ili metode i kad nije potreban eksplicitan konstruktor

Općenito	govoreći, što od navedenog vrijedi za lambda izraz?
a	u izrazu koji se nalazi desno od strelice "->" definiramo ponašanje metode iz funkcijskog sučelja
b	u izrazu koji se nalazi lijevo od strelice "->", uz argument obavezno moramo navesti tip argumenta kako bi prevoditelj znao o kojem se funkcijskom sučelju radi
С	može zamijeniti anonimnu klasu koja za cilj ima implementirati sučelje s jednom apstraktnom metodom
d	može zamijeniti anonimnu klasu koja za cilj ima implementirati određeno funkcijsko sučelje

Općenit	o govoreći, što od navedenog vrijedi za sučelje Consumer <t>?</t>
a	propisuje postojanje metode boolean test(T t)
b	za razliku od sučelja Predicate <t>, sučelje Consumer<t> se od Jave 8 nužno mora implementirati isključivo lambda izrazom</t></t>
С	propisuje postojanje metode T remap(T t)
d	predstavlja operaciju koja prihvaća jedan ulazni argument i ne vraća rezultat

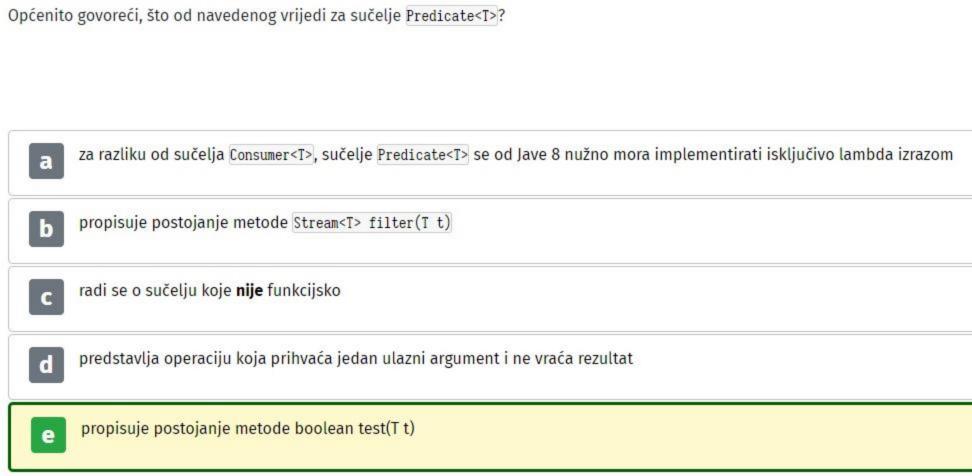
```
class Student {
    String id, name;
    public Student(String i, String n) {
        id = i:
        name = n;
public class Main {
   public static void main(String[] args) {
        Student pero1 = new Student("123456", "Pero");
        Student pero2 = new Student("123456", "Pero");
        System.out.print(pero1.equals(pero2) + ", ");
        System.out.println(pero1.equals(pero1));
```

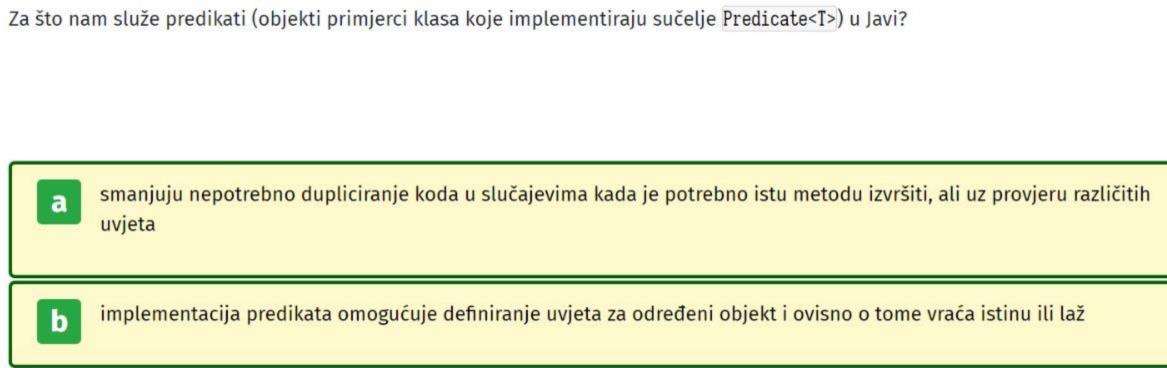
Sto ispisuje program?

Što vrije	di za statičke ugnježđene klase:
a	ovakva klase ne može imati konstruktor
b	instanca ovakve klase se može stvoriti na samo jednom mjestu u izvornom kodu
С	instanca ove klase ne može postojati bez instance vanjske klase
d	ovakva klasa nema ime
е	instanca ove klase može postojati bez postojanja instance vanjske klase

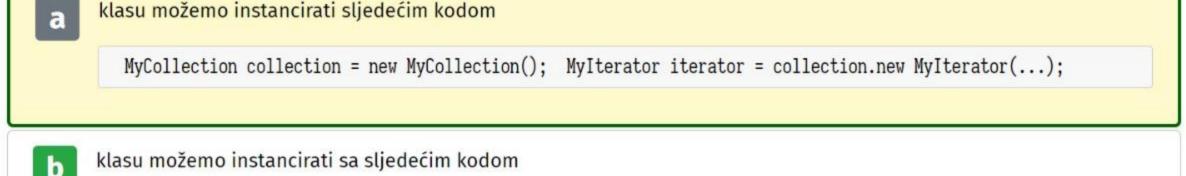
Ako bi k	lasa Kruska bila definirana s
pul	olic class Kruska implements Comparable <jabuka></jabuka>
tada bi	klasa Kruska morala imati metodu
a	Kompajler ne bi dopustio da klasa Kruska implementira Comparable <jabuka></jabuka>
Ь	int compareTo(Jabuka j)

Prirodni	poredak, odnosno prirodni komparator za neku klasu T definira se
a	definiranjem nekog komparatora koji se svodi na usporedbu članskih varijabli
Ь	implementacijom sučelja Comparable i metode compareTo(T other)
С	nadjačavanjem metode compareTo iz klase Object
d	pisanjem statičke metode comparable(T first, T second)
е	implementacijom sučelja Comparable i metode compareTo(T first, T second)



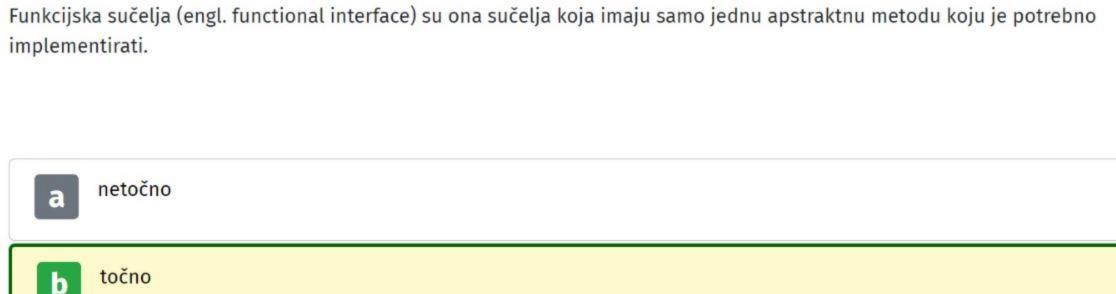


Ako implementiramo iterator pomoću statičke ugnježđene klase (MyIterator) koja se nalazi u klasi MyCollection onda za tu klasu vrijedi:



kroz konstruktor takve klase moramo poslati referencu na vanjsku klasu te se takva referenca mora spremiti u atribut

new MyCollection.MyIterator(...)



Koji od na	vedenih potpisa su potpisi metoda equals i hashCode iz klase java.lang.Object?
a	int hashCode()
b	int hashCode(Object obj)
С	boolean equals(Object obj1, Object obj2)
d	int equals(Object obj)
е	boolean equals(Object obj)