1. Koju apstraktnu metodu je potrebno nadjačati kod implementacije sučelja java.util.function.Predicate<T>?

* Boolean test(T t)

1. Koje su metode definirane sučeljem java.lang.Iterator<T>?

* Boolean hasNext();
* T next ();
* Reasource(); (valjda ?)
* Iterator<T> iterator();

1. Dokumentacija sučelja java.util.Collection<T> definira pojam opcionalnih metoda. Što to znači za razrede koji implementiraju to sučelje i neće podržati akcije koje provode te metode?

* Ti razredi trebaju implementirati takve metode na način da u tijelu istih izazovu iznimku UnsupportedOperationException

1. Dokumentacija sučelja java.util.Collection traži od razreda koji implementiraju to sučelje(ili neko iz njega izvedeno) da bi trebali podržati određen broj konstuktora. Od ponuđenih, o kojima se radi?

* Konstruktor koji prima referencu na neku drugu kolekciju te u konstruiranu kolekciju kopira sve elemente iz predane kolekcije
* Konstruktor koji ne prima niti jedan argument i stvara praznu kolekciju

1. Što od ponuđenog vrijedi za sučelje java.util.collection?

* Collection<T> extends Iterable<T>

1. Što od navedenog je istina za java.util.Collections?

* Java.util.Collections sadrži samo statičke metode
* To je klasa

1. Koja izjava vrijedi za java.util.LinkedList

* Lista elemente pohranjuje dinamički alocirajući nove čvorove za dvostruko povezanu listu

1. Za I/O tokove u javi vrijedi

* Tokovi su jednosmjerni
* Klijent u izlazni tok može isključivo zapisivati
* Tokovi se mogu ulančati korištenjem dekoratora

1. Ispravan način definiranja toka podataka iz datoteke je

* Tok = new FileInputStream(<Ime\_datoteke>)
* Tok = Files.newInputStream(<path\_do\_datoteke>)

1. Označite metode sučelja FileVisitor

* postVisitDirectory
* preVisitDirectory
* visitFile
* visitFileFailed

1. Znakovni tokovi unutar paketa java.io modelirani su apstraktnim razredima

* Writer
* Reader

1. Metode reverse, shuffle i sort dio su klase ili sučelja

* Collections

1. Ako riječi dodajemo u LinkedHashSet redoslijedom kojim su navedene u gornjoj rečenici, što ćemo dobiti kad sadržaj kolekcije ispišemo

* Sve isto osim što se riječi ne ponavljaju, znači ako se riječ već pojavila drugi put je neće biti

1. Ako riječi dodajemo u TreeSet redoslijedom kojim su navedene u gornjoj rečenici , što ćemo dobiti kad sadržaj kolekcije ispišemo

* Abacednim redom, bez ponavljaja riječi

1. Listu možemo stvoriti pozivom

* New ArrayList<>();
* Array.asList(„A“, „B“, „C“);
* New LinkedList();
* Array.asList(1,2,3);

1. Lista je uređena kolekcija koja

* Može sadržavati duplikate

1. Izvor i ponor okteta modelirani su

* Apstraktnim klasama InputStream i OutputStream

1. Mostovi između znakovnih tokova i tokova okteta su

* InputStremReader
* OutputStreamWriter

1. Most Između znakovnih tokova i tokova okteta je

* InputStreamReader

1. Koji od ponuđenih naziva predstavljaju klase koje implementiraju sučelje java.util.Set, a koje pripadaju javinom okviru kolekcija

* HashSet
* TreeSet
* LinkedHashSet

1. Implementacija sučelja java.util.Set okvira kolekcija koja osigurava da su elementi sortirani je:

* TreeSet

1. Implementacija sučelja java.util.Set iz Javinog okvira kolekcija koja osigurava da su elementi poredani redoslijedom dodavanja je

* LinkedHashSet

1. Javin okvir kolekcija nudi sljedeće implementacije sučelja java.util.List:

* ArrayList
* LinkedList

1. Kolekcija Map je parametrizirana s tipom ključa i vrijednosti.Što može biti korišteno kao ključ i vrijednost?

* Bilo koja klasa
* Bilo koje sučelje

1. Koje su karakteristike kolekcije Map?

* Svaki ključ ima pridruženu jednu vrijednost
* Mapa ne može pohranjivanja više istih ključeva
* Više ključeva može imati vrijednost

1. Što od navedenog je ispravno ako je map tipa Map<String, Integer>?

* For(String entry: map.keySet()){

//radi nešto

}

* For (Map.Entry<String, Integer> entry: map.entrySet()){

//radi nešto

}

1. Što sve od navedenog je ispravno ako je map tipa Map<String, Map<String,Integer>>?

* For(Map.Entry<String, Map<String, Integer>> entry : map.entrySet()){

//radi nešto

}

* For (Map<String, Integer> entry : map.values()){

//radi nešto

}

1. Ako je mapa definirana s Map<String, Integer> map ispravno inicijalizirana te je prazna u trenutku izvođenja sljedećih naredbi

Map.put(„Ana“, 5);

Map.put(„Ana“, 4);

System.out.println(map.size());

System.out.println(map.get(„Ana“));

Što će se ispisati na ekranu?

* 1,4

1. Ako je skup definiran s Set<String> set ispravno inicijalizirani te je prazan trenutku izvođenja sljedećih naredbi

System.out.println(set.add(„Ana“));

System.out.println(set.add(„Ana“));

Što će se ispisati na ekranu?

* True, false

1. Metoda vratiKolekciju() vraća ne-null referncu na neku nepromjenjivu kolekciju. Što je ponuđenog potrebno upisati na mjesto označeno podvlakama u kodu kako metoda m ne bi svojem pozivatelju propagilala iznimku nastalu uslijed poziva metode add u bloku try?

Public void m(){

Try{

Collection<String> kolekcija = vratiKolekciju();

Kolekcija.add(„Pero“);

System.out.println(kolekcija.contains(„Pero“));

}

Catch(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_){ }

}

* UnsupportedOperationException ex

1. Metoda vratiKolekciju() vraća referencu na neku kolekciju. Što treba upisati na mjesta označena u kodu kako bi program ispisao sve elemente kolekcije na zaslon?

Collection<String> kolekcija = vratiKolekciju();

For(\_\_\_(1)\_\_\_\_ : \_\_\_\_(2)\_\_\_\_\_){

System.ot.println(\_\_\_\_\_(3)\_\_\_\_);

}

* (1) String str, (2) kolekcija, (3) str

1. Napisali smo razred koji implementira sučelje Filename…..

Public boolen accept(File dir, String name) { ---KOD----}

* Return dir.isDirectory();

1. Što od navedenog vrijedi pri preslikavanju primjeraka razreda File i Path:

* Primjerci razreda File mogu se preslikati u primjerke razreda Path koristeći metodu toPath, a primjerci razreda Path mogu se preslikati u primjerke razreda File koristeći metodu toFile()

1. Ako se za obradu strukture direktorija oslonimo na metodu Files.walkFileTree(…), jedan od objekata koji je potrebno predati može biti objekt koji je (označite sve što je točno)

* Primjerak klase koja implementira FileVisitor
* Primjerak klase koja nasljeđuje SimpleFileVisitor

1. Ako se za obradu strukture direktorija oslonimo na metodu Files.walkFileTree(…), označite sve što je potrebno predati toj metodi kroz argumente:

* Objekt koji implementira sučelje FileVisitor
* Vršni direktorij iz kojeg kreće obilazak

1. Pretpostavimo da koristimo računalo s Windows OS-om i da NE POSTOJI datoteka na putanji „d:/tmp/readme.txt“. što će se dogoditi ako pokrenemo program koji sadrži sljedeću liniju koda:

File f = new File(„d:/tmp/readme.txt“);

* Naredba će se uspješno izvršiti, tj. Stvorit će se objekt tipa File bez obzira na to što datoteka ne postoji.
* Neće se stvoriti datoteka readme.txt

1. Što sve vrijedi za kolekciju tipa List

* Može se dohvatiti element na zadanoj poziciji
* Može se umetnuti element na zadanu poziciju
* Može se obrisati element na zadanoj poziciji
* Numeracija pozicija počinje od 0
* Elementi imaju svoju poziciju unutar liste

1. Ako se kod kreiranje novog FileInputStream objekta proslijedi kao parametar ime datoteke koja ne postoji, dogodit će se

* Bacit će se Iznimka FileNotFoundException

1. Nad objektima koji su primjerci razreda koji implementiraju sučelje DirectoryStream, koje sve metode sigurno možemo pozvati(jer postoje)?

* Iterator()
* Close()