TEST 1

1. Šta je rekurzija?

   - Pod rekurzivnim metodom se podrazumeva metod koji poziva sam sebe. Rekurzija omogućava znatno jednostavnije rešavanje pojedinih problema.

2. Šta podrazumeva rekurzivno rešavanje problema?

   - **Rekurzivno rešavanje problema** podrazumeva da definišemo programski deo za jedan element složene strukture, pa da ga onda višestruko pozivamo da to ponovi. Kako se pozivi vrše iz prethodno relizovanog programskog dela, to se rekurzija hijerarhijski ponavlja.

3. Šta je rekurzivni metod?

   - Pod **Rekurzivnim metodom** se podrazumeva metoda koji poziva sam sebe.

4. Šta je beskonačna rekurzija?

   - **Bekonačna rekurzija** je rekurzija koja se nikada ne završava.

5. Koliko puta se poziva metoda factorijal() u programu datom za klasu ComputeFactorijal?

   - Poziva se sve dok se parametar koji se prenosi u samom metodu i smanjuje za 1, ne smanji na vrednost 0.

6. Šta je Direktna rekurzija?

   - **Direktna rekurzija** je rekurzija kada metoda poziva sam sebe.

7. Šta je indirektna rekurzija?

   - **Indirektna rekurzija** je kada jedan metoda poziva drugi metoda a on oet poziva prvo metod.

8. Kako biste problem Fibonačijevog niza podelili na rekurzivne potprobleme?

   - Podelili bismo ga na dva potproblema:
     1) Određivanje brojeva fib(index - 2) i
     2) Određivaenj brojeva fib(index - 1).

9. Koliko će metoda fib biti pozvan u listingu koji određuje Fibonačijeve brojeve za fib(6)?

- 16 puta. Jer u opštem slučaju računanje fib(index) zahteva 2 puta više rekurzivnih poziva, nego što bi broj rekurzivnih poziva imalo izvršenje fib(index – 1).

10. Opišite karakteristike rekurzivnih metoda.

- Karakteristike su:
  1) Metod upotrebljava if-else ili swich iskaze koji vode do različitih slučajeva.
  2) Jedan ili više osnovnih slučajeva se koriste za zaustavljanje rekurzije.
  3) Svaki rekurzivni poziv smanjuje početni problem, dovodeći ga bliže do osnovnog slučaja, sve dok ne postane i on takav slučaj.

11. Koji je osnovni slučaj (base case) metoda isPŠalindrome() opisanog u materijalima? Koliko puta će ovaj metod bit pozvan za slučaj poziva isPalindrome(„abdxcxdba“)?

- Osnovni slučaj: “Provera da li je prva oznaka (character) jednaka poslednjoj oznaci stringa“. Ovo je ujedno i prvi potproblem a drugi potproblem je da li string dužine 0 ili 1. Metod će biti pozvan 5 puta.

12. Nacrtati ili napisati stek poziva (call stack) za slučaj da se funkcija poziva funkcija isPalindrome(„abcba“)?

- Poziva se 5 puta:
1) isPalindrome(“abcba”),
2) isPalindrome(“bcb”),
   isPalindrome(“abcba”),
3) isPalindrome(“c”),
   isPalindrome(“bcb”),
   isPalindrome(“abcba”),
4) isPalindrome(“bcb”),
   isPalindrome(“abcba”),
5) isPalindrome(“abcba”).

13. Šta je rekurzivni pomoćni mteod (recursive helper method)?

- Praksa pri rekurzivnm programiranju je da se definiše drugi metod koji prima dodatne parametre i pomaze glavnom rekurzivnom problemu. Takav metod je poznat kao **rekurzivni pomoćni metod**. Ovi metodi su korisni kod rekuzivnog rešavanja problema koji se javljaju sa stringovima i nizovima.

14. Kako biste problem Hanojskih kula podelili na rekurzivne potprobleme?

- Tako što stubove (podstabla) prebacimo pomoću sledećeg algoritma:
  1) Prebaciti stablo koje sadrži n-1 diskova sa A na B.
  2) Preostali (najveći) disk prebaciti sa A na C.
  3) Prebaciti podstablo sa B na C.

15. Koliko puta će metoda moveDiscs() biti pozvam za slučaj moveDiscs(5, 'A','B','C')?

- 31 put.

16. Koja od sledećih tvrdnji je tačna?

- Rekurzivni metodi zahtevaju više vremena i memorije za izvršenje nego nerekurzivni metodi.

17. Šta je repna rekurzija?

- Pod **Repnom rekurzijom** (tail recursion), podrazumeva se rekurzija koja nema zaostale operacije na čekanju a koje bi trebalo izvršiti po završetku jednog rekurzivnog poziva.

18. Kada biste koristili repnu rekurziju?

- Repnu rekurziju bih koristio kada treba na efikasan način da smanjim potrebnu veličinu steka.

19. Identifikovati primere ove lekcije koji koriste repnu rekurziju?

- Klasa ComputeFactorialTailRecursion ima pomoćni repni rekurzivni metoda za faktorijel po imenu factorial().

20. Koliko puta se poziva rekurzivna funkcija za rad sa Hanojskim kulama kada su na ulaznom štapu 4 diska?

- 15 puta.

21. Analizirati sledeći rekurzivni metod: public static long factorial(int n) { return n*factorial(n-1);}

- Metoda radi beskonačno što kao posledicu izaziva StackOverFlowError.

TEST 2

1. Šta omogućava primena generičkih klasa i metoda?

    - Omogućava da otkrijemo greške u vreme kompilacije programa umesto u vreme njegovog izvršenja.

2. Zašto se koriste generički tipovi?

    - Zato što generički tipovi (generici) kao parametri omogućavaju da ponovo koristimo isti kod sa drugim ulazima. Prednosti korišćenja su pre svega: Bolja provera tipa u vreme kompilacije, I Eliminacija konverzija tipa.

3. Šta je autoboxing?

    - **Autoboxing** je direktno dodeljivanje elemenata nekoj promenljivoj primitivnog tipa ako su elementi liste učaureni tipovi (wrapper types).

4. Koja je generička definicija za java.lang.Comparadle u Java API?

    - public java.lang.Comparable<E>

5. Ako kreirate objekat ArrayString sa stringovima upotrebom new ArrayList(), da li se konstruktor u klasi ArrayList može definicati kao: public ArrayList()?

    - Obično se konstruktor definiše da bude private. ?????

6. Da li generička klasa može da ima višestruke generičke parametre?

    - Generička klasa ili metod dozvoljava vam da specificirate dozvoljene tipove objekata sa kojima klasa ili metod može da radi.

7. Kako deklarišete generički tip u klasi?

    - Svaki put kada kreiramo novu instancu tipa ProstGenerik u zagradama < > moramo definisati tip odnosno klasu koja se odnosi na generički tip. Generički tip se definiše za klasu ili interfejs.

8. Kako deklarišete neki generički metoda? Kada aktivirate generički metod?

- Da bi deklarisali generički metoda, treba da postavimo generički tip <E> odmah posle ključne reči static u zaglavlju metoda:

public static <E> void print(E[] list);

Da bi se aktivirao (prizvao) generički metoda, treba navesti ime metoda sa stvarnim tipom u uglastim zagradama:

GenericMethodDemo.<String> print (strings);

9. Šta je ograničen generički tip (bounded generic type)?

- Ograničen generički tip je generički tip koji se može specificirati i kao pod-tip nekog tipa.

10. Za dati red int[] list = {1. 2. -1} da li možete da pozovete metod sort(list) upotrebom metoda sort() datog u listingu klase GenericSort?

- DA. Zato što metod sort() sortira niz sa bilo kojim tipom objekata.

11. Za dati red int[] list = {new Integer(1), new Integer(2), new Integer(-1)} da li možete da prizovete sort(list) upotrebom sort metoda u listing na slici 1.?

- Da, možemo.

12. Šta je sirovi tip? Zašto je sirovi tip nebezbedan? Zašto je sirovi tip dozvoljen u Javi?

- Generička klasa koja se koristi bez tipa parametra naziva se sirovim tipom (raw type). Nebezbedan je zato što mogu da izazovu grešku pri izvršenju programa, jer se ne mogu upoređivati objekti različitog tipa (npr. String i int).
Sirovi tip u je dozvoljen u Javi zato što se njihovom upotrebom obezbeđuje kompatibilnost unazad.

13. Prikaži sintaksu deklaracije promenljive ArrayList upotrebom sirovog tipa i dodeljuje mu sirovi tip ArrayList.

- Generic lista: lista = new GenericList <ArrayList>();

14. Da li je GenericStack isto što i GenericStack?

- Da

15. Šta je džoker sa ograničenom donjom granicom, ograničeni džoker i neograničeni džoker?

- Ako se podtipovi objekata koji su u upotrebi razlikuju, onda se koriste džokeri generičkih tipova. Džoker generičkih tipova ima 3 forme:
    1) ? - neograničeni džoker,
    2) ? extendsT – ograničeni džoker,
    3) ?superT – je džoker sa donjom granicom.

16. Šta je brisanje? Zašto se Java generici upotrebljavaju za vreme brisanja?

- Brisanje tipa je pristup u kome kompajler upotrebljava informaciju o generičkom tipu da bi kompajlirao kod, ali ga posle briše. Ovaj pristup omogućava da generički kod ostvari kompatibilnost unazad sa starim kodom koji upotrebljava sirove tipove.

17. Ako program upotrebljava ArrayList i ArrayList<String> da li JVM<Integer> opterećuje oba niza?

- Iako su to dva razlicita tipa u vreme kompilacije, smo klasa ArrazList se unosi u JVM za vreme kompilacije.

18. Da li mozete da kreirate instancu upotrebom new E() z agenerički tip E? Zašto?

- Ne može da se upotrebljava new E(), zato što se new E() izvršava za vreme rada programa, a tada generički tip E nije raspoživ, jer su generički tipovi raspoloživiv samo za vreme kompilacije.

19. Da li neki metod može da upotrebljava statički parametar generičke klase? Zašto?

- Statički parametar generičke klase nije dozvoljen, zato što sve instance (objekti) generičke klase imaju istu klasu u vreme izvršenja pa se statičke promenljive i metodi generičke klase dele od strane svih instanci.

20. Da li možete da definišete prilagođenu generičku klasu izuzetka? Zašto?

- To je nemoguće zato što bi u slučaju da to može, bilo moguće uhvatiti (catch) klauzulu za MyException<E>.
JVM treba da proveri izbačen izuzetak iz try klauzule da bi proverila da li odgovara tipu koji je specificiran u catch klauzulu. To je nemoguće, jer ta inf. nije raspoloživa u vreme izvršenja programa.

21. O kojim ograničenjima se treba voditi računa kada su u pitanju Generički tipovi?

- Ograničenja su:
1) Ne može da se kreira instanca (objekat) upotrebom parametra generičkog tipa.
2) Ne možete kreirati niz (array) upotrebom parametra generičkog tipa.
3) Parametar genričkog tipa neke klase nije dozvoljen u statičkom kontekstu.
4) Klase izuzetka ne mogu da budu generičke, tj. ne mogu se kreirati, hvatati (catch) ili izbacivati (throw) objekti sa parametrizovanim tipovima.
5) Ne možete primeniti generičke tipove se primitivnim tipovima.
6) NE moćete upotrebiti konverziju tipova ili koristiti objekte sa parametrizovanim tipovima.
7) Ne možete preopteretiti (overload) metoda sa formalnim tipovima argumenata svakog brisanja preopterećenja u isti sirovi tip.

TEST 3

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

TEST 4

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

TEST 5

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

TEST 6

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.