TEST 2

1. Objasniti zbog čega se klasa može smatrati apstraktnim tipom podataka?

   - Zato što znamo samo kako da koristimo njene metode I javne atribute a njena programerska implementacija je skrivena od nas kao korisnika.

2. Zašto je enkapsulacija (učaurivanje) važna paradigma objektno-orjentisanog programiranja?

   - Zato što ona omogućava sakrivanje implementacije (primene) komponenti klase kao što su atributi I metodi od korisnika.

3. Koju prednost nudi učaurivanje, a koja nije dostupna u proceduralnom programiranju?

   - Prednost učaurivanja je u tome što korisnik ne mora da zna kako je klasa stvarno programerski realizovana, već samo kako da je koristi (tj. da koristi njene atribute i metode).

4. Proceduralnim pristupom programiranju se obezbeđuje višestruka upotrebljivost softverskih komponenti?

   - TRUE

5. Kod proceduralnog programiranja, podaci i operacije su odvojeni, što zahteva ubacivanje podataka u metode?

   - TRUE

6. Šta je ispravno?

   - OOP postavlja podatke i operacije koje ih koriste u isti objekat.
   - OOP name omogućava da se implementacija sakrije od krajnjeg korisnika klase.

7. Ukratko objasniti pojmove Asocijacije, Agregacije i Kompozicije.

   - **Asocijacija**, je opšta veza koja opisuje aktivnosti između dve klase.
   - **Agregacija**, je veza vlasništva između dva oblika, to je specijalan oblik asocijacije.
   - **Kompozicija** je vrsta agregacije, takva da predstavlja neku vrstu ekskluzivnog vlasništva, što znači da agregirani objekat ne može biti korišćen (tj. biti u vlasništvu drugog korisnika), kao što je to slučaj kod agregacije.

8. Koja je razlika između Agregacije i Kompozicije?

- Razlika je u aspektu vlasništva. Kod Agregacije klasa komponenta može biti korišćena i od strane drugih klasa (vlasnika), dok kod kompozicije agregirana klasa može biti korišćena samo od strane te jedne klase vlasnika u čijem je vlasništvu.

9. Relaizacija između klasa Student I Predmet predstavlja …

- Agregaciju

10. Koje su razlike između omotačkih (wrapper) klasa i primitivnih tipova podataka?

- Razlika je u tome što klase omotači omogućavaju da se vrednot primitivnog tipa koji nije objekat može umotati u objekat.

11. Koje su prednosti korišćenja primitivnih tipova podataka a koje omotačkih klasa?

- Prednosti korišćenja primitivnog tipa je u tome što on može da se automatski konvertuje u objekat upotrebom odgovarajuće omotačke klase.

12. Koje od sledećih parsiranja je moguće izvesti?

- Integer.parseInt("32");

13. Koje od sledećih parsiranja nije moguće realizovati?

- Double-parseDouble(21,3);

14. Šta će biti rezultat parsiranja, Float.parseFloat("12.512222798111");

- 12.512223

15. U kojim slučajevima se koriste BigInteger i BigDecimal?

- Klase **BigInteger** i **BigDecimal** se mogu koristiti za predstavljanje celih ili decimalnih brojeva bilo koje veličine i preciznosti.

16. Na koji način se vrše aritmetičke operacije sa objektima tipa BigInteger i BigDecimal?

- Pomoću metoda: add, substract, multiply, divide, remainder.

17. Da li je moguće vršiti osnovne aritmetičke operacije između objekata tipa BigInteger i BigDecimal?

- DA

18. Napisati kod za sve načine kojima se može kreirati string…

- String noviString = new String(stringLiteral);
- String poruka = " 'Welcome to Java';
- char[] nizKaraktera = {'G', 'O', 'O', 'D'};
- String poruka2 = new String(charArray);

19. Koja je razlika kada se string kreira pomoću konstruktora ili definisanjem teksta poda navodnicima?

- String objekti su nepromenljivi, to znači da kada kreiramo string pomoću navodnika mi ne možemo taj string više da promenimo, već možemo samo da napravimo novi string i dodelimo ga istoj promenljivoj, čime prektično prebrišemo staru vrednost koju je imala ta promenljiva pre toga.

20. Koja je razlika kada se stringovi porede sa "==" operatorom a koja sa metodom "equals()"?

- Operator "==" poredi reference string objekata, a ne njihove sadrzaje, dok metoda "equals()" poredi vrednosti sadrzane u string objektima.

21. Svaki put kada se menja vrednost String-a, String dobije novu referencu …

- FALSE

22. Regularni izrazi se koriste …

- Za proveru da li se String uklapa u određeni šablon.

23. Ukoliko dva String-a imaju istu vrednost, a oba su kreirana pomoću konstruktora …

- Oba String-a ukazuju na istu memorijsku lokaciju (referencu),
- Rezultat poređenja će biti TRUE samo ukoliko se poređenje vrši sa operatorom provere jednakosti ("==").

24. Komanda "PHP je super. PHP je najbolji programski jezik".replace("PHP."JAVA"); vraća sledeći rezultat:

- "JAVA je super. JAVA je najbolji porogramski jezik".

25. Koja je prednost rada sa String-ovima u odnosu na rad sa nizom karaktera (char[])?

- U okviru klase String je obezbeđen veliki broj pomoćnih metoda koje nisu dostupne, kada je u pitanju niz karaktera.

26. Svaki put kada se menja vrednost String-a, String dobije novu referencu …

- TRUE

27. Regularni izrazi se koriste …

- Za proveru da li se String uklapa u određeni šablon.

28. Navesti prednosti korišćenja StringBulider i StringBuffer klasa?

- Prednosti su u tome što su one fleksibilnije od klase String. Možemo da ubacimo nov sadržaj u StringBuilder i StringBuffer objekte, dok u slučaju String objekata, jednom formiran String objekat ne može da se menja.

29. Koje su razlike između Klasa StringBuilder I StringBuffer?

- Razlika je u tome što su metodi za modifikaciju bafera u StringBuffer klasi sinhronizovani.

30. Koja od sledećih tvrdnji je netačna?

- StringBuilder i StringBuffer ne mogu da rade sa String-om većim od 256 karaktera.

TEST 3

1. Podklasa je podskup superklase. Da li je to tačno?

    - NE. Podklasa nije podskup superklase i ona sadrži više svojstava nego njena super klasa.

2. Koju ključnu reč koristite pri definiciji klase?

    - **extends**

3. Šta je jednostruko nasleđivanje?

    - To je slučaj u kome se dobijanje podklase vrši proširivanjem samo jedne klase, tj. jedna podklasa ima više superklasa.

4. Šta je višestruko nasleđivanje?

    - To je slučaj u kome se dobijanje podklase vrši proširivanjem nekoliko klasa, tj. jedna podklsa ima više superklasa.

5. Da li Java podržava višestruko nalseđivanje?

    - NE

6. Da li su privatni atributi superklase dostupni van nje?

    - Privatni atributi superklase nisu dostupni u njenoj podklasi. Oni mogu postati dostupni upotrebom javnih metoda.

7. Koja od ovih klasa se ne može nasleđivati?

    - final class A {}

8. Koja od ovih modifikacija najpribližnije određuje klasu naslednicu u odnosu na klasu koja se nasleđuje?

    - Proširivanje

9. Označite tvrdnje koje su tačne …

- Klasa A nasleđuje klasu B, to znači da je klasa A podklasa klase B.

10. Koja od ovih rečenica o službenoj reči SUPER je tačna?

- Službena reč SUPER može poslužiti za proširivanje konstruktora nasleđene klase.
- Službena reč SUPER može poslužiti za povezivanje enkapsuliranog metoda nasleđene klase.

11. Koja od navedenih tvrdnji je tačna?

- Metod se može preopteretiti u istoj klasi.

12. Objasniti pojam Polimorfizam.

- **Polimorfizam** znači da promenljiva super tipa može da predstavlja podtip objekta. On omogućava da se ista operacija superklase različito primenjuje u podklasama.

13. Navesti tri osnovna sidrišta objektno-orjentisanog programiranja.

- To su: 1. Učaurivanje, 2. Nasleđivanje, 3. Polimorfizam.

14. Šta je tačno od sledećih tvrdnji?

- Svaki primerak (objekat) podklase je istovremeno i primerak (objekat) njegove superklase, ali suprotno ne važi.
- Veza nasleđivanja omogućava da podklasa nasledi svojstva svoje superklase sa dodatnim svojstvima.

15. Šta je dinamičko vezivanje?

- Dinamičko vezivanje je slučaj kada možemo pozvati neki metod sa bilo kojim objektom hijerarhijskog lanca.

16. Objasnite razliku između uparivanja metoda i vezivanja metoda.

- **Uparivanje metoda** znači da deklarisan tip reference promenljive odlučuje koji će metod biti uparen u vreme izvršenja, a kompajler nalazi uparen metoda u skladu sa tipom parametara, brojem parametara I redosleda parametara u vreme izvršenja.
- JVM dinamički vezuje implementaciju metoda u vreme izvršenja, zavisno od stvarnog tipa promenljive. To je mehanizam dinamičkog **vezivanja metoda**.

17. Kako definišemo eksplicitnu konverziju tipa objekta?

- Referenca objekta se može konvertovati u drugu referencu objekta. Ovo se zove konverzija objekata (casting).

18. Čemu služi metod "instanceOf()"?

- Program koristi metod "instanceOf()" da bi proverio da li je izvorni objekat primerak ciljne klase pre nego što se izvrši konverzija tipova.

19. Čemu služi ArrayList?

- Klasa ArrayList služi za memorisanje objekata u formi liste.

20. Da li u jednoj ArrayListi mogu da se nadju dva objekta različitog tipa?

- Da (mada u skripti piše Ne)

21. Da li je moguće kreirati listu elemenata tipa int?

- Ne

22. Koji modifikator treba koristiti u klasi tako da klase iz istog paketa mogu da pristupe njenim članovima, ali da ne mogu da pristupe iz drugih paketa?

- private

23. Koji bi modifikator trebalo a koristite tako da klase iz različitog paketa ne mogu da joj pristupe, ali mogu podklase iz bilo kog paketa?

- protected

24. Uvek možete da konvertujete tipove objekta podklase u tipove superklase. Da li je to tačno ili nije?

- Uvek je mogućno konvertovati primerak podklase u promenljivoj superklase, jer primerak objekta podklase je uvek primerak (objekat) i svoje superklase.