TEST 1

1. Koje su osnovne razlike između objektno-orjentisanog i proceduralnog programiranja?

- **OOP**, je programiranje sa upotrebom objekata, gde se rešenje traži na prirodan način koji bi se primenio i u stvarnom životu, tj. kreiraju se klase objekata kojima se na prirodan način modelira problem koji se rešava.
  Proceduralno programiranje, je pristup u kome programer mora da identifikuje računarski postupak za rešavanje problema i da napiše niz programskih naredbi kojima se taj postupak realizuje.

2. Šta predstavlja objekat u objektno-orjentisanom programiranju?

- **Objekat** je entitet (realni ili apstraktni) koji ima svoj identitet i svojstva. On je instanca klase.

3. Koja je razlika između objekta i klase?

- Razlika je u tome što se Klasa definiše u tekstu programa, dok se Objekti te klase konstruušu posebnim operatorom tokom izvršavanja programa.

4. Navesti koja su tri osnovna svojstva objekata?

- Osnovna **svojstva objekata** su:
  1. Stanje objekta (opisano atributima),
  2. Ponašanje objekta (definisano metodama),
  3. Identitet objekta (svojstvo po kome se objekti razlikuju).

5. Iz čega se sastoji svaka Klasa?

- Sastoji se iz:
  - **Atributa**, koji opisuju svojstva i karakteristike svake klase.
  - **Metoda**, koji rade sa atributima klase.

6. Koja je razlika između statičkih i nestatičkih atributa klase?

- Razlika je u tome što Statički atributi (promenljive) ne menjaju vrednost a Nestatički atributi imaju različitu vrednost kod svakog objekta iste klase.

7. Koja je razlika između reference na objekat i samog objekta?

- Razlika je u tome što je Referenca adresa objekta i odnosi se na adresu memorijske lokacija objekta u hip memoriji, a Objekat je entitet i ima neke svoje karakteristike.

8. Koja je uloga konstruktora i zbog čega se on razlikuje od ostalih metoda?

- **Konstruktor** je metod koji stvara objekte svoje klase. On ima ime jednako imenu klase, parametre i modifikator pristupa.

9. Šta u objektno-orjentisanom programiranju predstavlja Enkapsulacija (Učaurivanje)?

- **Enkapsulacija** (učaurivanje) je objektno-orjentisano načelo koje nalaže da svi objektni atributi klase budu skriveni i deklarisani sa modifikatorom pristupa "private" i pristup vrednostima tih atributa iz drugih Klasa treba omogućiti samo preko javnih (public) metoda.

10. Čemu služi UML klasni dijaram?

- UML klasni dijagram (dijagram klasa) prikazuje skup Klasa, interfejsa i njihovih relacija. On opisuje strukturu sistema.

11. Koji su elementarni tipova veza između objekata?

- Postoji 5 elementarnih tipova veza (relacija) između objekata:
  1) **Asocijacija** – jedan objekat upotrebljava servise drugog,
  2) **Agregacija** – jedan objekat sadrži u sebi drugi objekat,
  3) **Kompozicija** – gde je vlasnik odgovoran za kreiranje ili uništavanje delova objekata,
  4) **Generalizacija** – gde podklasa ima sve karakteristike klase,
  5) **Usavršavanje** – je dodavanje prethodno nespecificiranih aspekata nekompletno specificiranom entitetu.

12. Objasniti pojam Apstrakcije?

- **Apstrakcija** je pojam koji se vezuje za Apstraktne klase, koje predstavljaju koncept objekata a ne stvarne realne objekte. **Apstraktna klasa** sadrži svojstva svih svojih podklasa ali nema svoje instance (objekte).

13. Objasniti polimorfizam?

- **Polimorfizam**, predstavlja tehniku korišćenja različitih Klasa iz iste hijerarhije klasa (familije), koju može koristiti bilo koji član familije klasa. On omogućava programeru da koristi bilo koju podklasu na mestu na kom se zahteva korišćenje superklase (obrnuto nije mogućno).

14. Objasniti nasleđivanje?

- **Nasleđivanje**, je veza tipa kao što je. Koristi se za modeliranje odnosa koji postoje između različitih ali vrlo sličnih Klasa, koje na ovaj način postaju delovi iste familije. Ovde "klasa-dete" nasleđuje sve atribute i metode "klase-roditelja" a ujedno može da sadrži i neke svoje dodatne attribute i metode.

15. Detaljno objasniti pojmove Klase, Objekta, Reference i opisati funkcije klazula Static i This?

1) **Klasa**, opisuje objekte sa sličnim karakteristikama i ponašanjem, ona je opšti opis nekog objekta.
2) **Objekat**, je entitet koji je instance klase i ima neke svoje karakteristike.
3) **Referenca**, je adresa objekta i predstavlja adresu memorijske lokacije objekta u Hip memoriji.
4) **Static**, označava statičku promenljivu (atribut) koja ne menja svoju vrednost u programu.
5) **This**, je ključna reč koja označava klauzulu kojom se kreira konstruktor koji ne prima argumete.

16. Šta je Java Class Library – JCL i šta ona sadrži?

- **JCL** predstavlja skup biblioteka koje sadrže predefinisane klase napisane u Java prog. jez. i koje pokrivaju šitok spektar problema. JCL se učitava u toku izvršenja programa.

17. Navedite neke od često korišćenih standardnih Java biblioteka?

- To su: Math, I/O, XML, Networking, … i sl.

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. Kako definišemo eksplicitnu konverziju tipa objekta?

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

17. 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.**

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.

TEST 4

1. Objasnite šta je to izuzetak i kako on nastaje?

- Izuzetak je objekat koji predstavlja neku grešku ili neki uslov koji sprečava normalno izvršenje programa. Izuzeci se stvaraju iz nekog metoda.

2. Objasnite kako se dobija informacija iz izuzetka?

- Objekat izuzetka sadrži informaciju o izuzetku. Možemo upotrebiti metode klase java.lang.Throwable da bi smo dobili informacije o izuzetku.

3. Objasnite kako funkcioniše Try-throw-catch blok?

- Princip rada: "Blok **try** sadrži metod koji se izvršava u normalnim okolnostima. Izuzetak izbačen sa **throw** naredbom se hvata u bloku **catch**, koji izvršava programski kod koji obrađuje izuzetak".

4. Navedite i objasnite tri glavne vrste izuzetaka …

- Tri glavne vrste izuzetaka su:
  1. **Sistemske greške** – se izbacuju od strane JVM i predstavljaju objekte klase Error.
  2. **Izuzeci** – su objekti klase Exception, koja opisuje greške prouzrokovane našim programom i spoljnim okolnostima. One se hvataju i obrađuju programom koji pravimo.
  3. **Izuzeci u fazi izvršenja** – su objekti klase RuntimeException, koja opisuje greške programiranja.

5. Objasnite Java model za rukovanje izuzetcima …

- Java model za rukovanje izuzetcima se zasniva na realizaciji 3 osnovne operacije, i to:
  1. Objava izuzetaka,
  2. Izbacivanje izuzetaka,
  3. Hvatanje izuzetaka.

6. Ako želite da se kompletan kod izvrši bez obzira na javljanje izuzetaka, koji klauzulu ćete koristiti?

- **finally**

7. Koji kod se izvršava u normalnim okolnostima?

- Kod obuhvaćen klauzulom **try**.

8. Objasniti kako Java relizuje ponovo izbacivanje izuzetaka?

- Java dozvoljava da obrađivač izuzetaka (iskaz throw ex) ponovo izbaci izuzetak, ako obrađivač ne može da obradi izuzetak ili ako želi da onaj koji poziva dobije informaciju o izuzetku.

9. Šta je korisnička Klasa izuzetaka?

- Ukoliko ni jedan od postojećih izuzetaka iz JAVA biblioteke ne odgovara našim potrebama, Java nam dozvoljava da definišemo sopstvene izuzetke kao podklasu klase Exception ili neke njene podklase (npr. IOException).

10. Opišite Java klasu Throwable, njene podklase i tipova izuzetaka.

- Java izuzetak je primerak klase koja se dobija od java.lang.Thorwable. Java obezbeđuje određena broj definisanih Klasa izuzetaka, kao što su:
    1. Error,
    2. Exception,
    3. RuntimeException,
    4. ClassNotFoundException,
    5. NullPointerException,
    6. AritmeticException.

TEST 5

TEST 6