TEST 1

1. Šta je Java?

- **Java** je objektno-orijentisani jezik opste namene, koji se može izvršavati na svim platformama, a ujedno, Java je i tehnologija koja sadrži sadrži pakete klasa i sadrži i okruženje za izvršenje programa (JRE).

2. Kako se podešava operativni sistem za pisanje Java programa?

- Da bi se na nekom OS-u mogao pisati Java program, potrebno je instalirati Javu i izvršiti podešavanje sistemskih varijabli kako bi Java bila dostupna sa bilo koje lokacije na samom računaru.

3. Koja poboljšanja donosi Java u odnosu na C/C++?

Poboljšanja su:
- Izbačeni su pokazivači, a rezervisanje i oslobađanje memorije je prepušteno Javi,
- Izbačeno je višestruko nasleđivanje i uvedeni su interfejsi,
- Ubačene su dodatne provere da bi se izbegle nenamerne greške,
- Nizovi su implementirani kao objekat a ne preko pokazivača.

4. Šta je JDK?

- Java Development Kit (**JDK**) predstavlja skup programskih alata za razvoj Java aplikacija.

5. Šta je JRE?

- Java Runtime Environment (**JRE**) predstavlja deo Jave koji je potreban za pokretanje i izvršavanje Java programa.

6. Pod kojim nazivom mora biti snimljena datoteka sa izvornim Java kodom?

- **Datoteka** sa izvornim kodom mora biti snimljena pod istim imenom pod kojim je snimljena i klasa koja se nalazi u toj datoteci.

7. U kojim slučajevima se javljaju sintaksne greške u kodu?

- **Sintaksne greške** u kodu se mogu javiti pri prevodjenju izvornog programa (koda) i to najčešće u slučajevima kada nismo ispoštovali proceduru kreiranja i pisanja programa.

8. Šta su logičke greške u programu?

- **Logičke greške** su neočekivane greške i javljaju se u radu programa. One se manifestuju u samom radu programa i još se nazivaju bagovi (Bug).

9. Šta su komentari i kako se koriste u Java programima?

- **Komentari** pomažu programerima da objasne program koji pišu, ostalim korisnicima da razumeju kako program radi, a koriste se i za dokumentovanje programa.

10. Šta je definisano programskim blokovima?

- **Programski blokovi** definišu grupu instrukcija metoda (u slučaju metoda) ili grupu podataka i metoda (u slučaju klase).

11. Kako se prevodi i pokreće Java program sa komandne linije?

- **Prevođenje** se vrši pomoću prevodioca javac, u bajtkod koji se nalazi u .class fajlu, pozivanjem komande javac u komandnoj liniji. **Pokretanje** programa se vrši pozivanjem komande java u komandnoj liniji i navođenjem imena fajla bez ekstenzije .class.

12. Koje su prednosti programiranja primenom integrisanog razvojnog okruženja IDE?

Prednosti korišćenja IDE-a su:
1. Editovanje koda, kompletiranje koda skraćenicama i prečicama, syntax highlighing, pomoć-help,
2. Pretraživanje koda, refaktorisanje koda,
3. Kompajliranje i izvršavanje, integrisan debugger,
4. Korišćenje čarobnjaka (wizards) koji olakšavaju komplikovane zadatke,
5. Kreiranje izvršnih i instalacionih paketa.

13. Koja su tri najčešće korišćena menija NetBeans IDE? Objasnite ih ukratko?

Meniji su:
1. Meni **File** - za formiranje novih i otvaranje postojećih projekata, fajlova, i za ostale aktivnosti u vezi sa fajlovima od kojih se sastoji Java aplikacija,
2. Meni **Edit** - ima istu funkciju za rad sa tekstom programa kao istoimeni meniji u ostalim grafičkim okruženjma (Undo, Redo, Copy, Paste, Find, Replace).
3. Meni **Run** - za izvršavanje prevedenog programa. I za pokretanje debager-a radi otklanjanja grešaka ili testiranja.

14. Koje NetBeans panele koristimo prilikom razvoja Java programa? Objasniti ih ukratko.

Paneli su:
1. **Panel projekata** - sadrži kartice Projects, Files i Services u kojima se hijerarhijski prikazuju projekti, datoteke i servisi koji su bitni za program.
2. **Panel navigatora** - prikazuje detalje stavki koje su izabrane u jednoj od tri pomenute kartice panela projekata.
3. **Radni panel** - prikazuje sadržaj datoteka koje su izabrane u jednoj od kartica panela projekata. Sadržaj otvorenih datoteka u radnom panelu se može menjati.
4. **Izlazni panel** - prikazuje sve važne poruke tokom prevođenja programa, kao i ulazno-izlazne podatke samog programa tokom njegovog izvršavanja.

15. Kako se kreira, prevodi i pokreće Java program primenom NetBean IDE-a?

Koraci su sledeci:
1. **Kreiranje** - Izaberem opciju File -> New Project i popunim podatke o projektu.
2. **Prevodjenje** - Izaberem opcije Run -> Run File u glavnom prozoru NetBeansa, ako se rezultat prevođenja završava porukom BUILD SUCCESSFUL, to znači da je prevođenje bilo uspešno i da se program može izvršiti.
3. **Pokretanje** programa - se vrši biranjem opcije Run -> Run Project

16. Navedite najpoznatija Java okruženja:

Najpoznatija **Java okruženja** su:
1. **NetBeans** – besplatno razvojno okruženje, sadrži sve neophodne elemente potrebne za razvoj aplikacija, može se proširivati dodacima.
2. **Eclipse** – besplatno razvojno okruženje, sadrži sve neophodne elemente potrebne za razvoj aplikacija, može se prošitivati dodacima.
3. **IntelliJ IDEA** – komercijalno okruženje, pogodno za sve tipove aplikacija, ima ogroman broj dodataka, podršku za Javu i druge jezike koje proizvode bajtkod.

17. Šta predstavlja **Java projekat**?

- Osnovna jedinica rada u nekom radnom okruženju je **projekat**. Projekat odgovara, jednom **Java programu** i sastoji se od njegovih izvornih datoteka, pridruženih informacija o tome kako ih treba prevesti i izvršiti, kao i od ostalih tehničkih detalja o putanjama, dokumentaciji i slično. Sve ove podatke radno okruženje smešta u direktorijumu projekta (sa poddirektorijumima) koji se pravi na osnovu imena projekta.

TEST 2

1. Šta čini elemente programskog jezika Java?

- **Program** čini niz deklarativnih i izvršnih naredbi, a njih čini niz leksičkih simbola. Leksički simboli ili tokeni predstavljaju nizove znakova. **Programski jezik Java** sadrži skup znakova (character set) koji čine:
1. mala i velika slova alfabeta,
2. deset decimalnih cifara,
3. veći broj znakova interpunkcije

2. Šta su tipovi podataka?

- **Tip podataka** definiše veličinu i organizaciju podataka, opseg mogućih vrednosti i skup operacija koje se mogu obaviti nad tim vrednostima. Najmanja količina podataka koja može da se adresira je jedan bajt.

3. Kako se dele tipovi podataka?

Tipovi podataka se dele u dve glavne grupe:
1. **Primitivni (prosti) tipovi** podataka - su tipovi podataka koji su unapred "ugrađeni" u jezik i postoje kod svih prog. jez. To su: celi brojevi, realni brojevi, logičke vrednosti, znakovi alfabeta.
2. **Složeni (klasni) tipovi** podataka (objekti) - su novi tipovi podataka koje programer sam definiše ili koristi već definisane. Sastoje se od nekoliko elemenata koji mogu da se nezavisno obrađuju. To su: nizovi i klase.

4. Šta su identifikatori?

- **Identifikator** predstavlja ime (naziv) promenljivih, tj. memorijskih lokacija, datoteka, metoda, atributa, objekata i klasa.

5. Šta su ključne reči?

- **Ključne reči** su reči rezervisane za specijalnu namenu u Javi i one se ne mogu korisiti za identifikatore koje daje programer.

6. Po čemu je specifičan tip String?

- **String** je predefinisana klasa u Java biblioteci koja omogućava formiranje teksta koji čini skup znakovnih tipova (tj. Char). On nije primitivni tip podataka, tj. stringovi su objekti unapred definisane klase String. String je zapravo referentni tip.

7. Šta su promenljive u Java programima?

- **Promenljiva** je ime lokacije u glavnoj memoriji koja koristi konkretan tip podatka za skladištenje određene vrednosti.

8. Šta su Konstanate u Java programima?

- **Konstanta** je promenljiva koja ne može promeniti vrednost tokom izvršavanja programa.

9. Šta su i zbog čega su značajni komentari u Java programima?

- **Komentar** predstavlja tekst na prirodnom jeziku u programu koji objašnjava delove programa drugim programerima. Komentari su značajni zato što drugi programeri mogu pomoću njih razumeti delove koda koje oni nisu pisali, a i sam programer koji je pisao taj kod se posle izvesnog vremena može podsetiti šta je hteo da uradi određenim kodom. Značajni su i za izradu projektne dokumentacije.

10. Koje vrste komentara postoje?

Postoje **tri vrste komentara**:
1. **Komentar u jednoj liniji** - je kratak komentar koji se piše u jednom redu teksta. Označava simbolom // i obuhvata tekst do kraja tog reda.
2. **Komentar u više linja** - je duži komentar koji se prostire u više redova. Počinje simbolom /*, obuhvata tekst u više redova i završava se simbolom */.
3. **Dokumentacioni komentar** - počinje sa /** a završava se sa */. Služi za pisanje dokumentacije direktno unutar samog koda

11. Koji identifikator je tačan od dole navedenih?

- Tačni su: Test, miles, $r

12. Koji od ovih iskaza je tačan, poštujući konvenciju koje se pridržavaju Java programeri radi bolje čitljivosti programa?

Tačni su **iskazi**:
1. Imena konstanti se sastoji od svih velikih slova.
2. Imena promenljivih i metoda počinju malim slovom.

13. Promenljiva može biti deklarisana kao int a kasnije je moguće promenti deklaraciju u tip double?

- FALSE

14. Koje od navedenih reči predstavljaju rezervisane reči u Javi?

- true, public, class, double.

15. Šta od navedenog predstavlja ispravno napisan komentar u Javi?

- // komentar, /* komentar */, /** komentar */

16. Šta od navedenog predstavlja ispravno definisanu konstantu u Javi?

- final double PI=3.14159;

17. Šta je klasa?

- **Klasa** je tip podatka koji korisnik može sam da definiše i koji je složen jer sadrži druge podatke različitih tipova. Ona opisuje računaru koje podatke svi objekti klase imaju i šta takvi objekti mogu da urade.

18. Koji elementi čine jednu klasu?

Klasa sadrži sledeće **elemente**:
1. atribute ili polja (promenljive),
2. metode (procedure) koje poseduju svi objekti te klase.
Klase praktično definišu šablon kako izgledaju pojedini objekti tih klasa.

19. Šta je objekat?

- **Objekat** je podatak čiji je tip definisala klasa koja ga stvara. Objekat predstavlja entitet iz stvarnog života sa svojim jedinstvenim identitetom, i predstavljen je skupom podataka koji predstavljaju jednog činioca objektno-orijentisanog modela.

20. Šta je omotačka klasa (wrapper)?

- **Omotačka klasa** sadrži "umotan" primitivni tip i time ih pretvara u objekte i omogućava da se nad njima sprovedu operacije kao i nad drugim objektima

21. Šta predstavlja paket u Java programima?

- **Paket** u programskom jeziku Java je kolekcija klasa koje su namenjene jednoj (određenoj) vrsti posla i te klase čine funkcionalnu celinu.

22. Kako se vrši unos i prikazivanje podaka u konzolnim Java programima?

- Klasa Scanner omogućava unos podataka u Java aplikacju. A objekat System.out sadrži metode za prikazivanje vrednosti na ekranu.

23. Da li će sledeći iskaz dati tačan rezultat: **System.out.println ("25/4 is " + 25/4);** ?

- FALSE (NE)

24. Da li različiti tipovi numeričkih vrednosti se mogu koristiti zajedno u jednom proračunu?

- FALSE (NE)

25. Koja će biti vrednost promenljive **a** na kraju: **int a=3; a++;** ?

- biće 4

26. Šta su operatori?

- **Operatori** predstavljaju operacije koje se izvršavaju nad operandima dajući određeni rezultat.

27. Koje vrste operatora poznajete?

Operatori se prema raznim kriterijumima **mogu podeliti na**:
1. Unarne operatore,
2. Binarne operatore,
3. Ternarne operatore,
4. Aritmetičke opreatore.

28. Koja je razlika između klase i objekta?

Razlika između objeka i klase je u sledećem:
1. **Objekat** je instanca klase i predstavlja stvarni FIZIČKI entitet koji programer kreira pomoću tastature. Pri tome alocira se memorija potrebna za njegovo čuvanje.
2. **Klasa** je šablon po kome se kreiraju objekti i može da predstavlja skup sličnih objekata. Klasa je LOGIČKI entitet, što znači da se pri kreiranju klase ne alocira memorija za njeno čuvanje.

29. Da li se Stringovi u Javi razlikuju od nizova znakova u nekom drugom programskom jeziku?

- Da, razlikuju se. U prog. jez. Java stringovi nisu nizovi karaktera, već objekti tipa String, pa se pojedinačnim karakterima unutar stringa ne može pristupiti preko njegovog indeksa već je potrebno upotrebiti pristupnu metodu, poput: charAt. Za razliku od Jave, u prog. jez. C++, stringovi se tretiraju kao niz karaktera i svakom karakteru u tom stringu se moze pristupiti preko njegovog indeksa, kao što bi se to uradilo u običnom nizu.

TEST 3

1. Šta su relacioni operatori?

- **Relacioni operatori** su operatori koji upoređuju dve vrednosti. Promenljive čije se vrednosti upoređuju nazivaju se operandi, a relacioni operator poredi vrednosti ta dva operanda i daje konačan rezultat. Kao rezultat oni daju logičke vrednosti TRUE ili FALSE.

2. Koje relacione operatore poznajete? Prikažite tabele operacija za njih …

- >, <, >=, <=, ==, !=

3. Pretpostavljajući da je x = 1, odredite rezultate sledećih logičkoh operacija: (x>0), (x<0), (x!=0), (x>=0), (x!=1).

- Rezultati su: true, false, true, true, false.

4. Kako se vrši poređenje objekata?

- Poređenje se vrši prema sledećoj logici: Dve objektne reference (promenljive ili konstante) su jednake ako ukazuju na isti objekat, što znači da ukazuju na istu memorijsku lokacija.

5. Objasnite metod equals() …

- Metod **equals()** upoređuje vrednosti atributa istog tipa i daje vrednost TRUE ako njihovi atributi imaju iste vrednosti.

6. Navedite opšti oblik naredbe if. Kada se izvršava naredba if ili blok naredbi vezan za naredbu if?

- Opšti oblik naredbe if je:

```
if ( logickiIzraz )
        naredba;

ili

if ( logickiIzraz ) {
        naredba1;
        naredba2;
        …
        naredbaN;
}
```

- Naredba ili blok naredbi vezan za naredbu if se izvrsava zavisno od toga da li je vrednost datog logickog izraza tacno (TRUE) ili netacno (FALSE).

7. Naredba ili blok naredbi, pridružen naredbi grananja if, izvršiće se ...

- samo u slučaju kada je logički izraz naredbe if tačan (TRUE).

8. Kada se izvršava naredba ili blok naredbi vezan za klauzulu else iz if-else strkture?

- Deo slozene if-else naredbe se preskace ako je uslov zadovoljen, tj. istinit (TRUE). Ako uslov nije zadovoljen (FALSE), izvrsavaju se naredbe iz else dela

9. Naredba ili blok naredbi, pridružen naredbi else, u najjednostavnijem if-else iskazu, izvršiće se …

- Ako je logički izraz uz if netačna (FALSE).

10. Ukoliko je neophodno realizovati višestruko grananje u programu koristićete iskaz:

- if – else if – else

11. U iskazu if – else if – else je tačan prvi logički izraz (uz if), i tada …

- naredba, tj. blok naredbi uz if se izvršava, ostale else-if i else naredbe se preskaču.

12. Kada ćete koristiti if else naredbu?

- if else naredbu ću koristiti kada treba da prvo izvršim proveru nekog zadatog uslova, a potom na osnovu ispunjenosti datog uslova, želim da izvršim jednu ili drugu naredbu (ili blok naredbi).

13. Kada se izvršava naredba ili blok naredbi vezan za klauzulu else if?

- Naredba ili blok naredbi vezan za klauzulu else if se izvršava ako je logički izraz uz if koje prethodi toj klauzuli netačan (FALSE).

14. Kod javljanja većeg broja grananja, gde je uslov grananja određen vrednošću neke promenljive ili rezultatom nekog izračunavanja, koristićete naredbu:

- switch

15. Objasnite ulogu naredbe break u višestrukom grananju.

- **Break** komanda prekida dalje izvrsavanje case dela i završava izvršavanje switch naredbe.

16. Kada se koristi naredba grananja switch?

- **Switch** naredba se koristi za uslovno izvršavanje izraza na bazi vrednosti celobrojnog izraza. U nekim slučajevima u praksi upravo switch naredba prirodnije održava logiku višestrukog grananja.

17. Šta je uslovni operator **?** i kako se on koristi?

- **Uslovni operator ?** (ili operator izbora) je ternarni operator koji se koristi umesto iskaza if. Uslovni operator ? omogućava skraćeno pisanje programa u odnosu na pisanje programa sa if-else naredbom, jer vraća jednu od dve moguće vrednosti.

18. Uslovnim operatorom **?** menja se iskaz …

- if

19. Kada bi ste koristili grananje if?

- **Grananje if** se koristi pri izboru jednog od dva alternativna niza naredbi za izvršenje zavisno od toga da li je vrednost datog logičkog izraza tačno ili netačna.

20. Kada bi ste koristili grananje switch?

- **Switch grananja** se koriste kada je potrebno održati logiku višestrukog grananja. Sama switch naredba se koristi za izvršavanje naredbi na bazi vrednosti celobrojnog izraza. Iskaz switch može da donese odluku samo na osnovu celobrojne vrednosti, dok se u iskazu if mogu koristiti različiti uslovi.

TEST 4

1. Objasnite rukovanje brojačem u while petlji.

- While petlja koristi, tzv. kontrolnu promenljivu (**brojač**) najčešće tipa int, i to i u uslovu i u telu petlje. Ovakve petlje predstavljaju poseban tip petlji, tzv. petlje sa brojanjem. Svaki prolaz kroz telo petlje se naziva iteracijom. Brojač povećava vrednost za 1 posle svakog prolaza petlje, tj. posle svake iteracije.

2. Kako glasi opšti oblik while petlje?

- Opšti oblik je: while (logickiIzraz) naredba;

ili

```
while (logickiIzraz){
    naredba1;
    …
    naredbaN;
}
```

3. Dati kratko objašnjenje principa rada while petlje…

- Izvršavanje **while** naredbe se izvodi tako što se najpre izračunava vrednost logičkog izraza u zagradi. Ako je ta vrednost FALSE, odmah se prekida izvršavanje while naredbe. Ako je TRUE, najpre se izvršava naredba ili niz naredbi u bloku, a zatim se ponovo izračunava logički izraz u zagradi i ponavlja isti postupak.

4. Navedite ključnu razliku između petlji while i do-while…

- Kod do-while petlje uslov za ponavljanje petlje se proverava na kraju petlje, za razliku od while petlje.

5. Kako glasi opšti oblike petlje do-while?

- Opti oblik je:

```
do
    naredba;
while(logickiIzraz);
```

ili

```
do{
    naredba1;
    …
    naredbaN;
}while(logickiIzraz);
```

6. Dajte kratko objašnjenje principa rada do-while petlje...

- Kod do-while petlje se najpre izvršava telo petlje, tj. naredba ili niz naredbi u bloku, zatim se izračunava vrednost logičkog izraza u zagradi. Ako je ta vrednost FALSE prekida se izvršavanje do-while naredbe i nastavlja se normalno izvršavanje programa od naredbe koja sledi iza reči while. Ako je TRUE, izvršavanje se vraća na početak tela petlje i ponavlja se isti postupak od početka

7. Objasnite upravljanje brojačem u for petlji?

- **Brojač** (count) se podešava na početku rada petlje. Inicijalizacija se odvija samo jednom. Uslov petlje se stalno ocenjuje, da bi se videlo da li telo petlje treba da se ponovo izvrši. Izraz petlje se izračunava uvek na kraju tela petlje.

8. Gde se nalazi uslov za zaustavljanje petlje for?

- Nalazi se u kontrolnom delu prilikom definisanja for petlje.

9. Objasnite kako je moguće izađi iz petlje pre njenog definisanog završetka?

- Uslov petlje proverava se pre izvrsavanja naredbe. Ukoliko uslov nije ispunjen, izlazi se iz petlje.

10. Kako se prekida tekuća iteracija petlje?

- Može da se prekine naredbom break.

11. Objasnite opšti oblik for petlje?

- Opšti oblik je:

```
for (inicijalizacija; logickiIzraz; zavrsnica){
                                        naredba;
}
```

12. Dajte kretko objašnjenje principa rada for petlje…

- U kontrolnom delu for petlje se obično nekoj promenljivoj daje početna vrednost, a u zavrsnici se ta vrednost uvećava ili smanjuje za odredjeni korak. Izmedju u kontrolnom delu proverava se logickiIzraz i petlja se izvršava sve dok je taj uslov ispunjen.

13. Kako se koriste naredbe break i continue u ugnježdenim petljama?

- Ove naredbe se mogu koristiti u ugnježdenim petljama radi prebacivanja kontrole izvršenja na kraj (sa break), odn. na početak petlje (sa continue). **BREAK** prekida dalje izvršenje petlje i šalje izvršenje na prvu liniju izvan koda petlje, **CONTINUE** prekida dalje izvršenje petlje i šalje izvršenje na početak petlje.

14. Šta predstavljaju ugnježdene petlje?

- **Ugnježdene petlje** su petlje koje mogu da se postave i unutar neke već postojeće petlje. Kod ovih petlji je pravila da se uvek prvo izvršava krajnja unutrašnja petlja, pa onda ona u kojoj se ova nalazi i tako redom.

15. Kako biste koristili i kako biste upravljali beskonačnom petljom?

- **Bekonačnu petlju** developeri koriste u slučajevima kada je potrebno održavati rad nekog pozadinskog procesa u aplikaciji. To su obično neki procesi koji vrše razne provere stanja (npr. servera), kopiranje/ažuriranje (backup) korisnikovih učitanih (upload) podataka, ... koriste se i u game develpomentu i sl.

TEST 5

1. Po čemu se razlikuju **aritmetički operatori**?

- Razlikuju se po tome što neki od njih rade sa jednim a neki sa dva operand. U tom smislu, operatori se mogu podeliti na unarne i binarne. Takođe, razlikuju se i po svom prioritetu, kao i po tome što obavljaju različite aritmetičke operacije.

2. Navedite i objasnite najčešće korišćene operatore…

- Najčešće korišćeni operatori su:

1. + unarni plus,
2. – unarni minus,
3. * množenje,
4. / deljenje,
5. % deljenje sa ostatkom,
6. + operator sabiranja,
7. – operator oduzimanja.

3. Po čemu se razlikuje ++i od i++ ?

- Razlikuje se po tome što: ++i znači povećanje pre upotrebe,
  i++ znači povećanje nakon upotrebe.

4. Po čemu se razlikuje - - i od i - - ?

- Razlikuje se po tome što: - - i znači smanjenje pre upotrebe,
  i - - znači smanjenje posle upotrebe.

5. Objasniti ulogu klase Math.

- **Klasa Math** se koristi za rešavanje matematičkih problema uz pomoć prog. jez. Java. Java ima mnogo f-ja zu klasi Math. I ona za svaku f-ju obezbeđuje po jedan metod, a ovo su static metodi, što znači da se mogu aktivirati upotrebom imena klase Math.

6. Šta podrazumeva polimorfizam pojedinih metoda klase Math?

- **Polimorfizam** označava metode koji u različitim kontekstima daju različite rezultate, iako se pozivaju na isti način. Pojedini metodi klase Math su polimorfni, jer mogu dati različite rezultate u zavisnosti od parametra koji se koriste u pozivu ovih metoda.

7. Šta je char?

- U prog. jez. Java, se za predstavljanje pojedinačnih znakova koristi znakovni tip podataka koji se naziva **char**. Ovaj tip može da predstavlja mala i velika slova, cifre, znakove interpunkcije i kontrolne znake.

8. Kako se prikazuju **znakovne konstante** u Javi?

- Predstavljaju se uokvirene jednostrukim navodnicima. To je bilo koji znak koji se može uneti sa tastature.

9. Šta je **String**?

- **Klasa String** je predefinisana Klasa u Java biblioteci koja omogućava formiranje teksta koji čini skup znakovnih tipova, tj. char.
  Objekat String predstavlja niz znakova (tip char) kojim se definiše neki tekst. Objekat nije primitivni tip podataka.

10. Po čemu se razlikuje String od char u Javi?

- Tip **char** predstavlja smo jedan znakovni tip u Javi. Da bismo predstavili ceo tekst, koristi se String tip podataka. **String** je predefinisana Klasa Java biblioteke koja omogućava formatiranje teksta koji čini skup znakovnih tipova.

11. Navesti i objasniti osnovne operacije sa Stringovima u Javi.

- **Osnovne operacije** su:

1. **concat()** – za spajanje dva String-a.
2. **toLowerCase()** – vraća novi niz koji sadrži samo male oznake.
3. **toUpperCase()** – vraća novi niz koji sadrži samo velike oznake.
4. **substring()** – izdvaja/iseca deo niza oznaka iz datog niza oznaka.
5. **length()** – vraća broj oznaka iz datog niza … i moge druge.

12. Zašto Klasa Math sama poziva svoje metode a ne njeni objekti?

- Zato što su svi metodi ove klase statički metodi, koji se mogu pozvati bez upotrebe nekog objekta, za razliku od metoda instance koji se ne mogu pozvati bez upotrebe nekog objekta.

TEST 6

1. Kako definišete metod?

   - **Metod** jeste potprogram u Javi koji se sastoji od niza naredbi koje se koriste za rešavanje nekog zadatka.

2. Šta podrazumeva definicija metode?

   - **Def. metode** podrazumeva definisanje zaglavlja metoda koje sadrži sve potrebne informacije neophodne za pozivanje metoda, tj. sadrži: ime metode, tip i imena parametara metoda, tip vrednosti koju metoda vraća kao rezultat i modifikator koji određuje dostupnost tj. vidljivost metoda.

3. Kako se poziva metod (opšti način)?

   - Pomoću: **imeMetoda(listaArgumenata);**

4. Nabrojati načine na koje se sve poziva metod?

   - Metodi se mogu pozivati na **3 načina**:
   1. Opšti način: imeMetode(listaArgumenata);
   2. Preko klase: ImeKlase.nazivMetode(listaArgumenata);
   3. Preko objekta: NazivObjekta.nazivMetode(listaArgumenata);

5. Da li se iz tela jedne metode može pozvati neka druga metoda?

   - DA

6. Koja je korist (prednost) od korišćenja metoda?

   - **Korist** se ogleda u tome što metodi omogućavaju da se složeni program podeli u manje delove koji se mogu lakše razvijati i održavati.

7. Šta je signatura metoda?

   - **Signatura** (potpis) metode, sastoji se od dve komponente koje se koriste pri deklaraciji metode a to su ime metode i tipovi i imena parametara koji se u toj metodi koriste.

8. Šta je **Parametar metode**?

   - To je vrednost koja se prosleđuje metodi. Metode obrađuju prosleđene parametre.

9. Šta je argument metode?

- **Argumenti** metode su parametri preko kojih se unose vrednosti u metod. Za svake argument potrebno je navesti i tip podatka koji koristi svaki argument.

10. Koji je povratni tip metoda main?

- void

11. Da li je TRUE ili FALSE da poziv metoda sa povratnim tipom return je iskaz za sebe, ali pozivanje metoda koji vraća neku vrednost ne može biti iskaz za sebe?

- FALSE

12. Šta je lokalna promenljiva?

- **Lokalne promenljive** su promenljive koje se definišu unutar samog metoda.

13. Šta je opseg važenja lokalne promenljive?

- To je oblast unutar domena (bloka) metoda u kome su deklarisane i van njega njima se ne može pristupiti.

14. Šta su modifikatori u metoodu?

- **Modifikatori** određuju karakteristike metoda. To su rezervisane reči u Javi.

15. Nabrojati neke od modifikatora u Javi.

- Razlikujemo sledeće modifikatore:
    1. public,
    2. private,
    3. protected,
    4. default.

16. Šta je preopterećenje (overloading) metoda?

- **Preopterećenje** (**overloading**) metoda nastaje kada dva ili više metoda u klasi imaju isto ime a različite listu argumenata.

17. Da li je dozvoljeno da definišete dva metoda koji imaju isto ime ali različite tipova parametara?

- DA

18. Da li je dozvoljeno da definišete dva metoda u klasi, koji imaju isti naziv i listu parametara, ali različitu vrednost koju vraćaju ili imaju različite modifikatore?

- NE

19. Kako se vrši apstrakcija metoda?

- **Apstrakcija** metoda se vrši tako što se odvaja upotreba metode od njene implementacije (izrade). Objekat koji poziva metod moguće je koristiti neznajući kako je on implemetiran (izrađen).

TEST 7

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

TEST 8

1.

2.

3.

4.

5.

6.

7.

8.