

- Mejl adresa za kontakt i materijal sa vežbi: asss.informatika@gmail.com
(potrebno je da svaki student/studentkinja pošalje mejl sa imenom, prezimenom, brojem indeksa, predmetom i grupom kojoj pripada kako bi bili registrovani na Google Drive nalog)

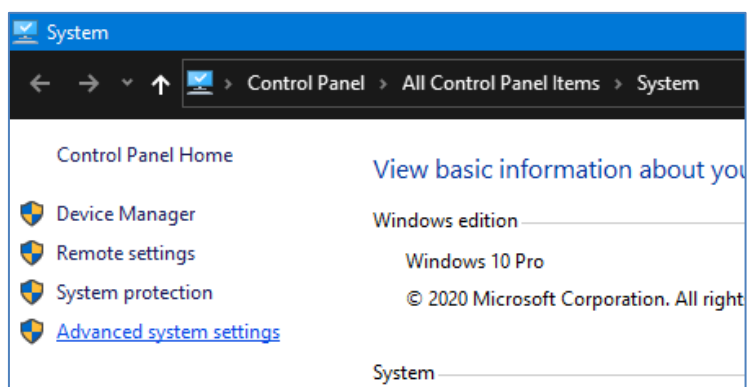
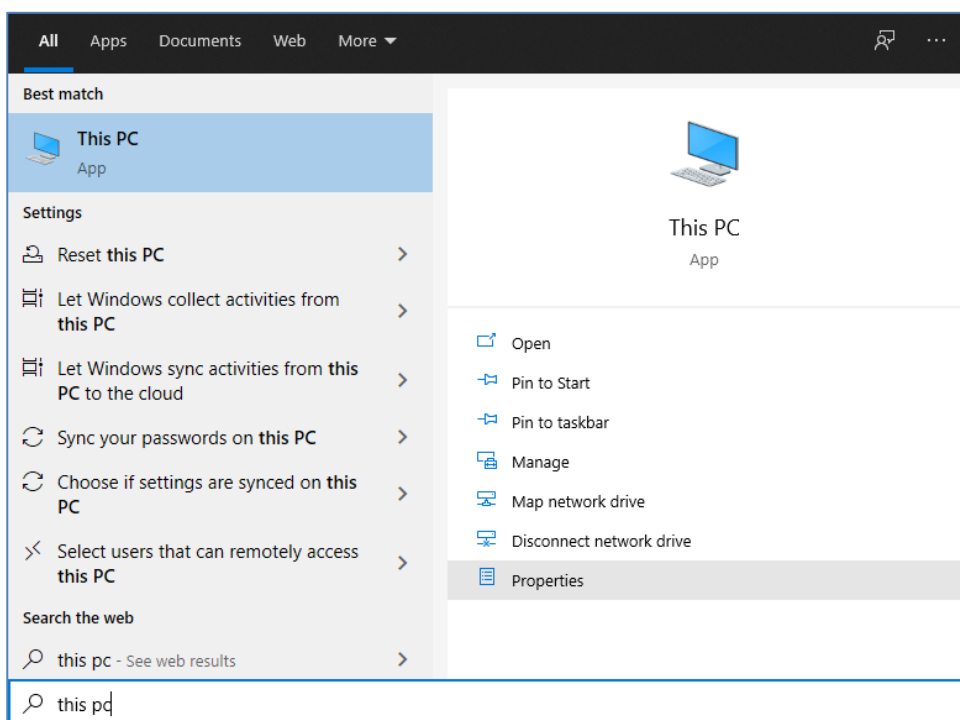
• Da biste bili u mogućnosti da pišete kod u Java programskom jeziku potrebno je:

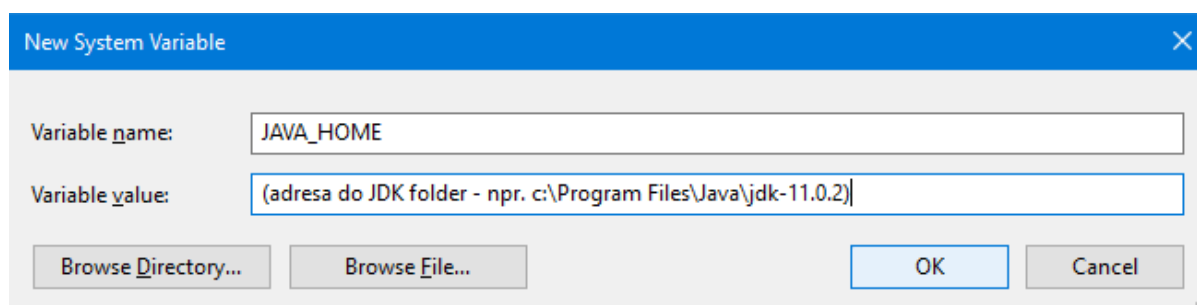
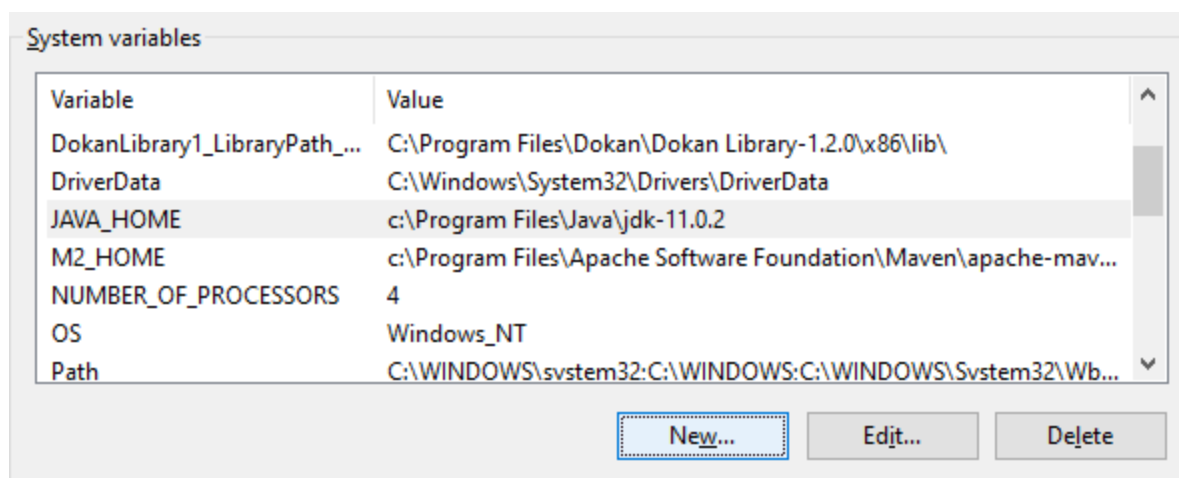
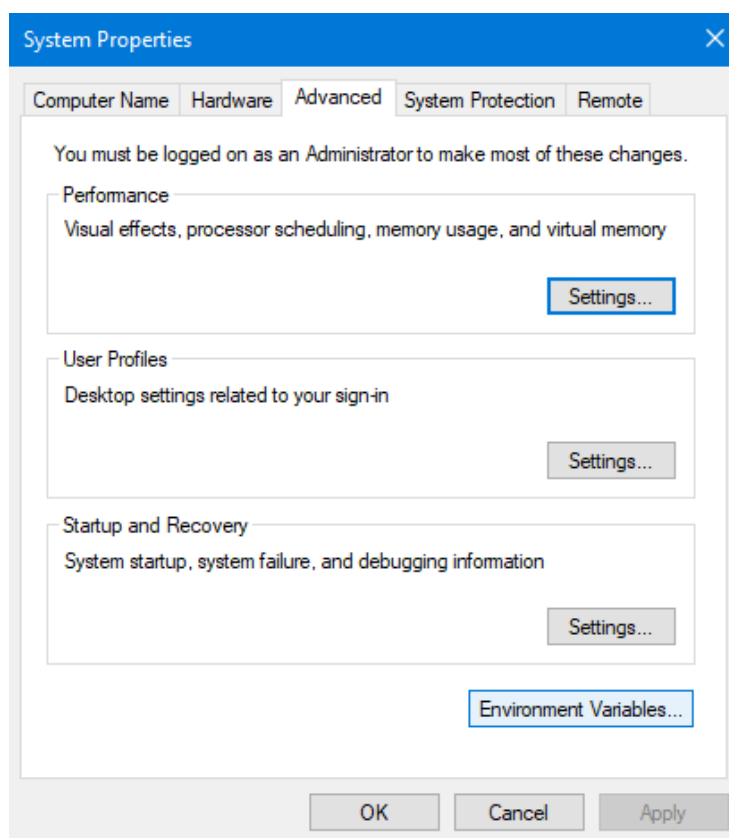
- instalirati Java Development Kit (**JDK**) i
- podesiti sistemske varijable (**Environment Variables**)
- instalirati text editor ili **IDE**.

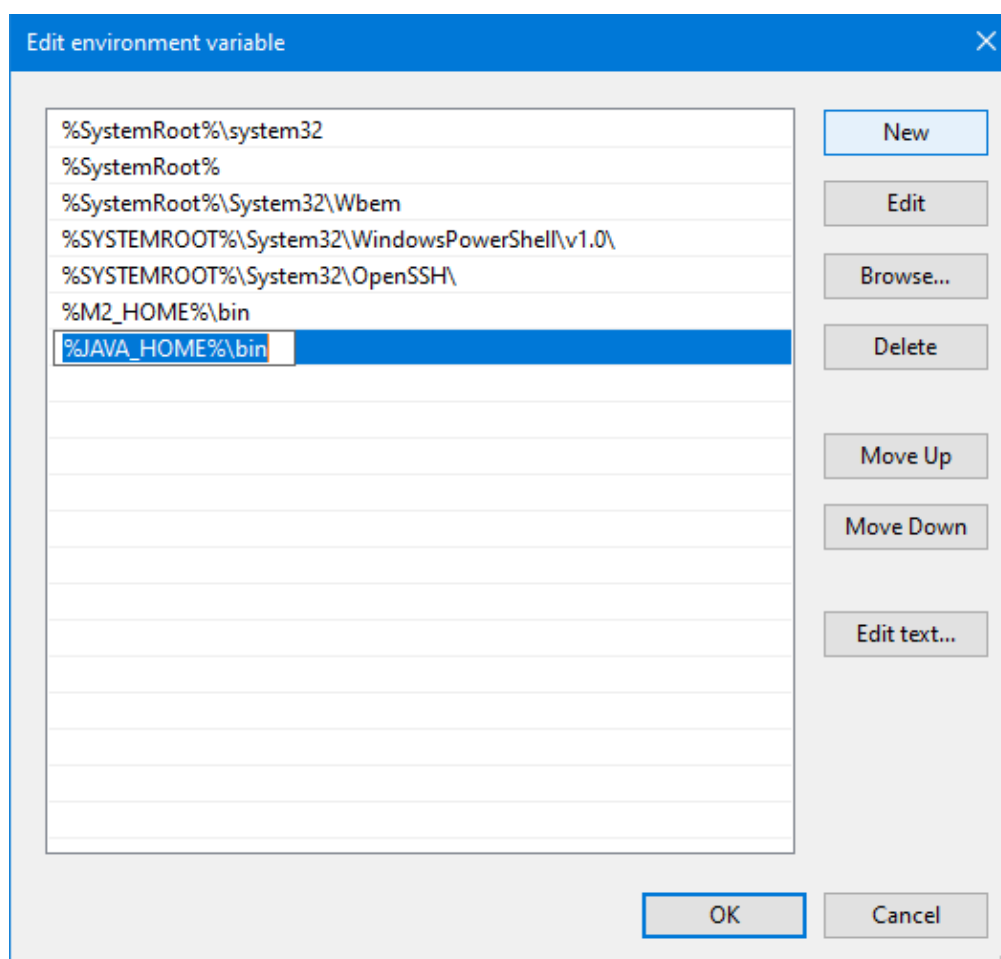
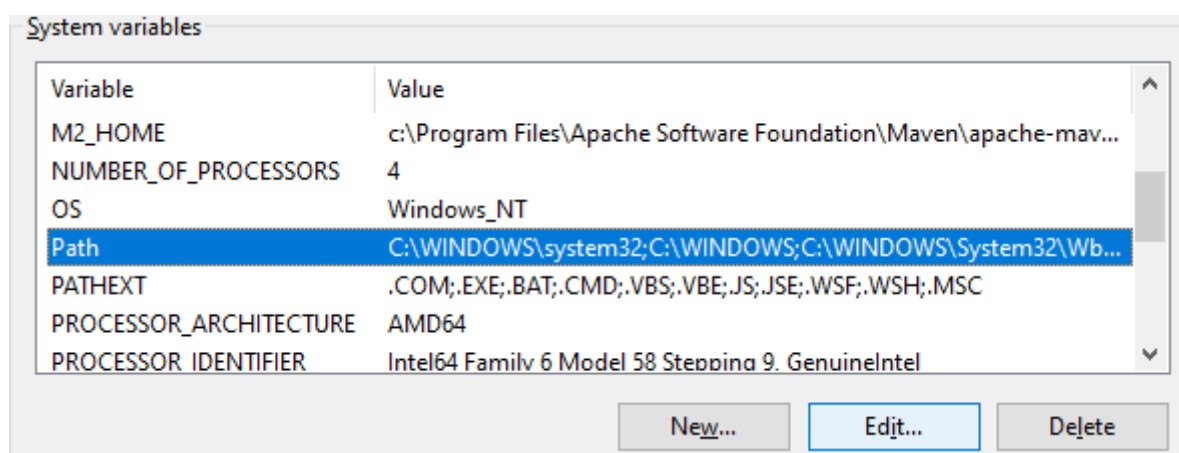
○ **JDK**

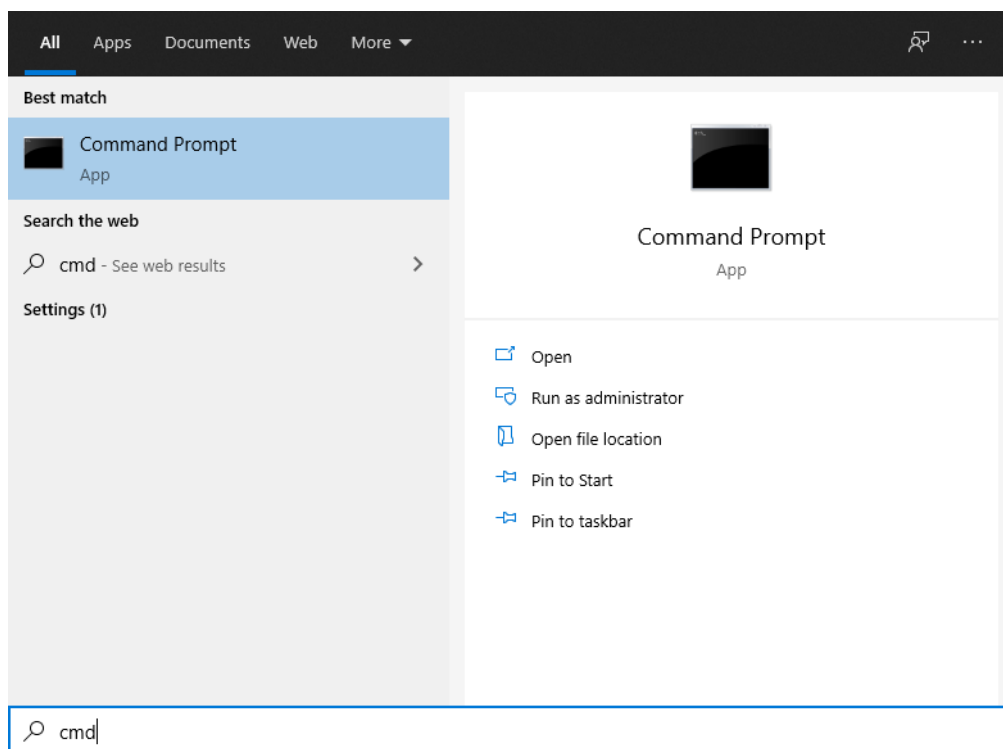
- <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>
- <https://jdk.java.net/java-se-ri/11>

○ **Enviroment Variables**







A screenshot of a Windows Command Prompt window. The title bar reads 'C:\> Command Prompt'. The window content shows the following text:

```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Aleksandra>java -version
openjdk version "11.0.2" 2019-01-15
OpenJDK Runtime Environment 18.9 (build 11.0.2+9)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.2+9, mixed mode)

C:\Users\Aleksandra>javac -version
javac 11.0.2

C:\Users\Aleksandra>
```

- **Razvojna okruženja (IDE): IntelliJ IDEA, Eclipse, NetBeans**
(na školskim kompjuterima je instaliran IntelliJ IDEA Ultimate – licenca se plaća, nastavne potrebe u potpunosti zadovoljava IntelliJ IDEA Community – besplatan)

Link za download IJ IDEA: <https://www.jetbrains.com/idea/download/#section=windows>
(odabirom kartice se može promeniti OS)

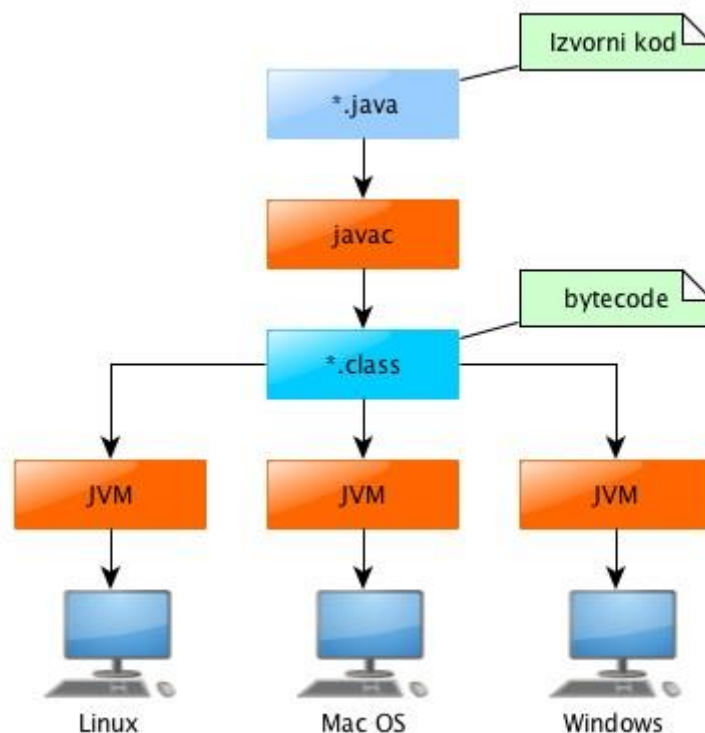
Demo za celokupni set up (IJ IDEA CE): <https://www.youtube.com/watch?v=EMLTOMdIz4w>

Java

Java je objektno orijentisani programski jezik koji se koristi za razvoj softvera koji se može izvršavati na različitim operativnim sistemima standalone i mobilnih uređaja (mašina).

Proces prevođenja (kompajliranja) je proces u kome se izvorni kod (razumljiv čoveku) napisan u jeziku višeg nivoa (Java, C++, ...) prevodi u format instrukcija koje procesor uređaja "razume". Programi koji vrše ovaj proces zovu se **prevodioci** (engl. *compilers*).

Java virtualna mašina (JVM) je softverski dodatak koji omogućava izvršavanje programa pisanih u Javi na bilo kom uređaju, nezavisno od njegovog operativnog sistema, čime obezbeđuje Javina univerzalnost primene i kompatibilnost sa svim uređajima/mašinama.



Ovu funkcionalnost programa pisanih u Javi obezbeđuje postupak obrade (prevodjenje i interpretiranje) Javinog izvornog koda.

Naime, za razliku od drugih (konkurentnih) programskih jezika, čiji se izvorni (čoveku razumljiv) kod direktno prevodi u mašinski (procesoru razumljiv) niz instrukcija, što ih čini direktno zavisnim od softversko-hardverske postavke same mašine, proizvod prevodjenja Javinog izvornog koda – eng. *source code* (***.java**) je Javin **bajt kod** - engl. *bytecode* (***.class**) koji predstavlja niz instrukcija za JVM (koji je sistemski zavistan).

Time je obezbedjena nezavisnost Javinih programa u odnosu na softversko-hardversku postavku masine, koja prvobitno mora imati na sebi instaliranu JVM.

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

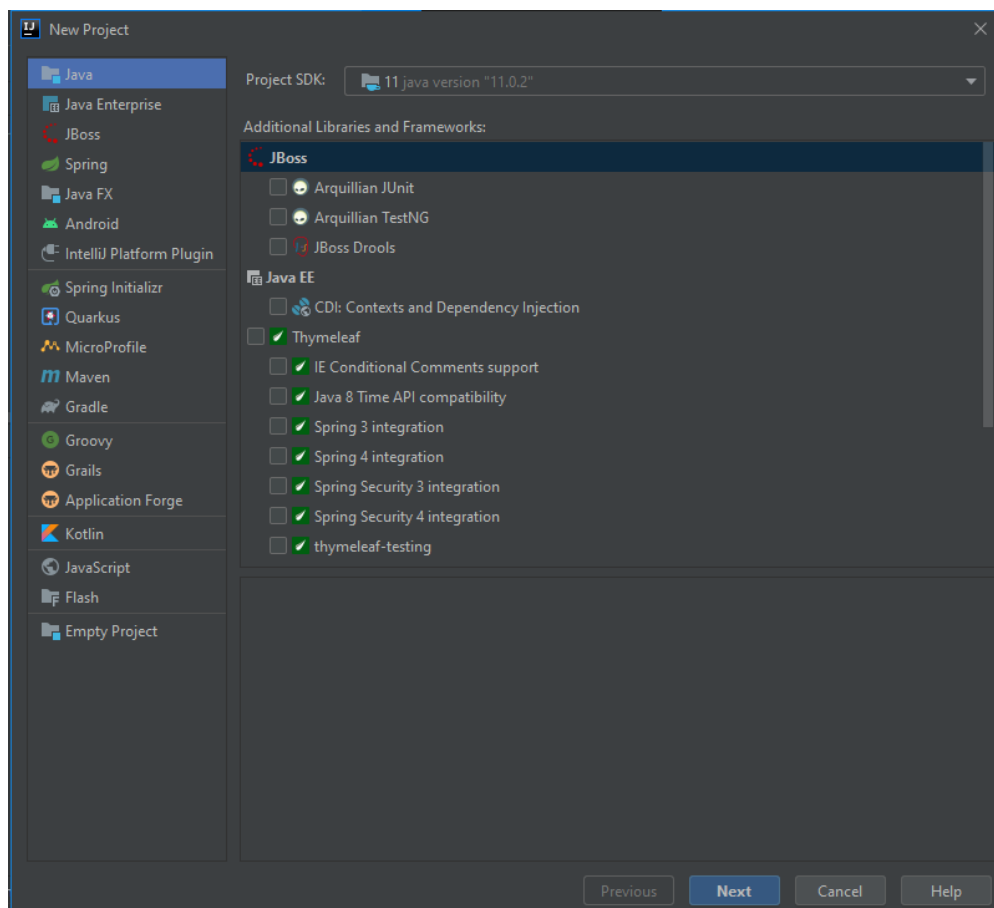
Java Development Kit (JDK) predstavlja skup programskih alata za razvoj Java aplikacija.

- Par uvodnih napomena o strukturi koda i IntelliJ IDEA razvojnom okruženju

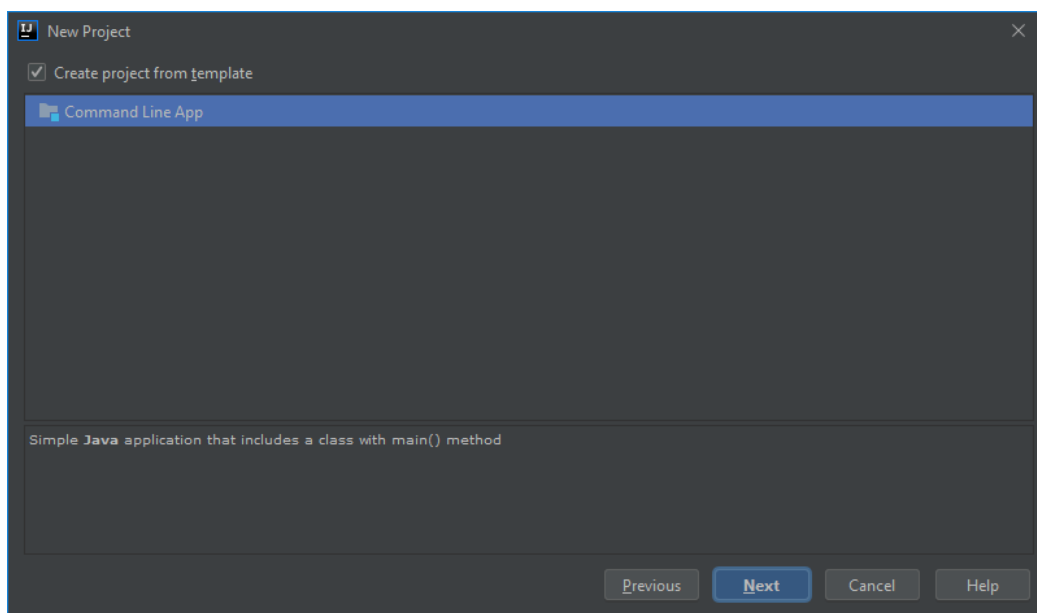
- Termini koji se često koriste:
program/aplikacija/projekat, paket, klasa, tip, objekat, polja/osobine (svojstva), metodi/ponašanje (funkcije), main metod - „main()“, iskaz, blok iskaza, operator tačka, zagrade (oble, uglaste, vitičaste, izlomljene), komentari, refactor, konvencije imenovanja (kamilja i zmijska).
- Najčešće korišćene celine IDE-a (razvojnog okruženja):
Text editor, Project, Structure, Run, Debug, kao Terminal i Maven

Primer 1 – Kako kreirati projekat i šta je Command Line App template

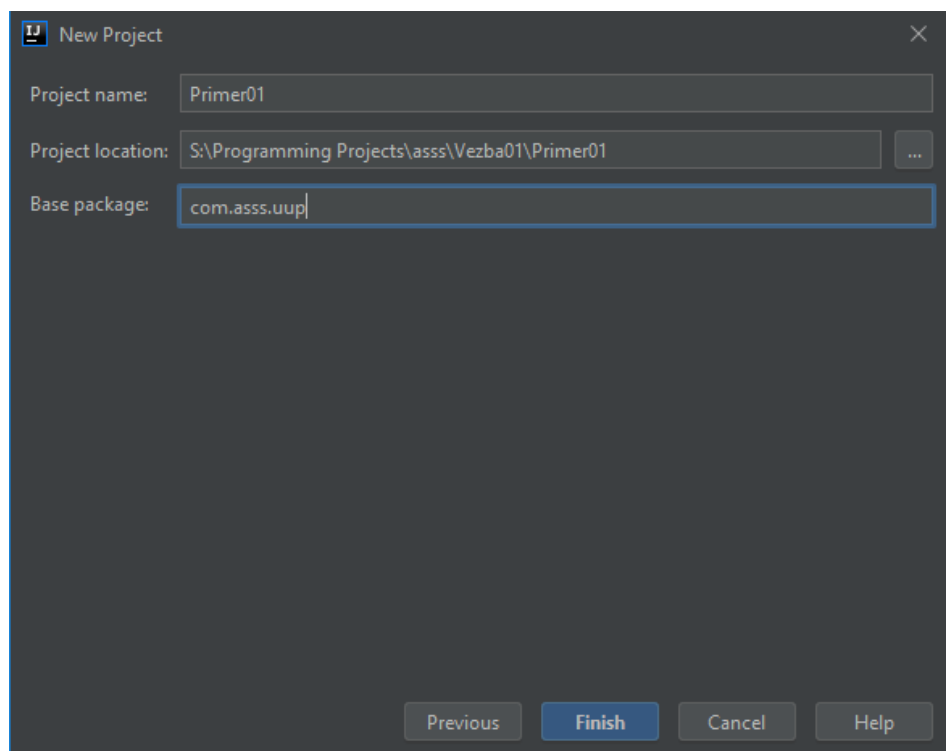
1. U zavisnosti od toga da li je okruženje podignuto ili ne, projekat se može napraviti na dva načina:
 - a. Prilikom pokretanja IDE-a sa „pozdravnog“ prozora (ukoliko nema zapamćenih, odnosno aktivnih projekata) i
 - b. Iz samog IDE-a (pomoću komandi **File > New > Project...**), nakon čega se prikazuje prozor sa projektima koji se mogu razvijati shodno verziji okruženja.



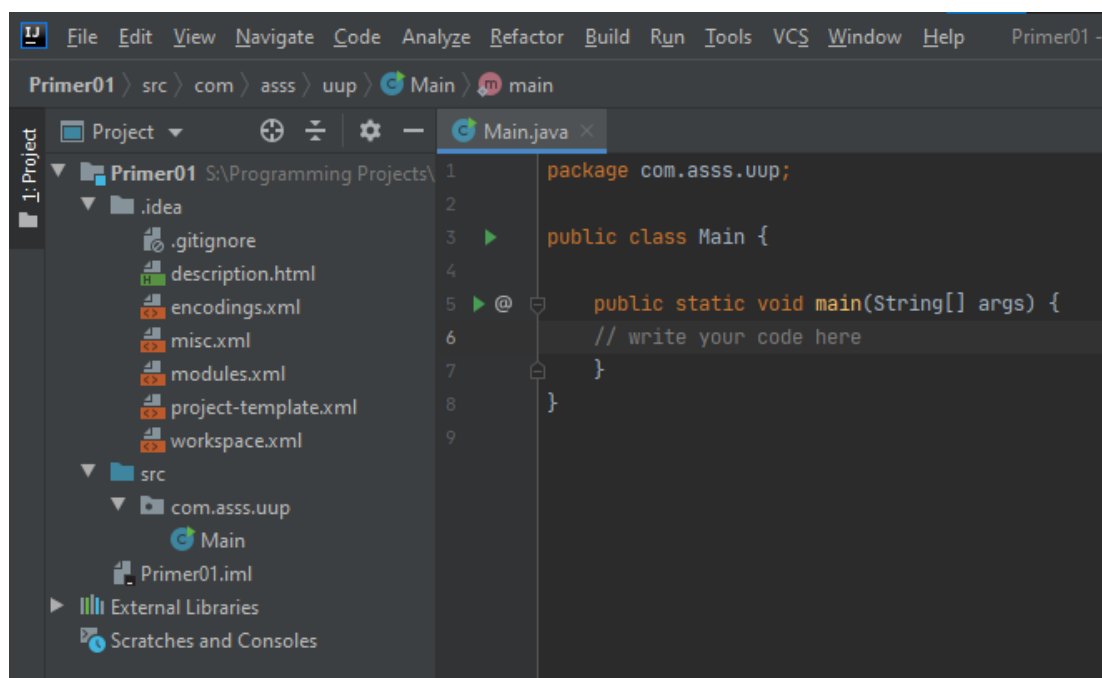
2. Uzimajući u obzir gradivo koje se obrađuje, svi projekti koje ćemo razvijati pripadaju kategoriji Java (bez ikakvih dodatnih biblioteka i radnih okvira).
Pre nego što se pređe na sledeći korak ,komandom „Next“, potrebno je odabrati verziju Java JDK-a na vrhu prozora iz padajućeg menija „Project SDK“ (software development kit).
3. Nakon prelaska na novi korak kreiranja projekta otvoriće se prozor koji će ponuditi predefinisane šablone shodno odabranom projektu i dodacima (u našem slučaju to će biti samo šablon pod nazivom „Command Line App“ koji predviđa upotrebu main() metoda – njegova upotreba će ubrzati pisanje koda).



4. Prelaskom na naredni i završni korak dolazimo do dela gde se definišu ime projekta, njegova adresa (na HDD-u) i naziv osnovnog paketa.
Ime projekta je ono ime po kojem će okruženje prepoznavati projekat i po kom će se moći, pomoću IDE-a, pristupiti projektu.
Adresa projekta predstavlja lokciju projekta na HDD-u, tj. njegovu apsolutnu putanju (obuhvata nazive svih foldera od particije na kojoj se projekat nalazi do samog foldera u kojem je projekat smešten, uključujući i njihova imena – particija i parent folder).
Osnovni paket je opcija koja je jedino prisutna u IntelliJ IDEA razvojnom okruženju kao podesiva, a razlog tome je pristup koji se koristi u praksi – u realnom svetu programiranja. Reč je zapravo o konvenciji (pravilu) koje programeri koriste, a to je da se svi paketi, od osnovnog naniž, pišu obrnutim redosledom u odnosu na domen (npr. asss.informatika@gmail.com >>> com.gmail.informatika.asss). Poštovanje ovog pristupa nije obavezno, ali je dobra praksa i značajno olakšava snalaženje programerima, naročito prilikom razvoja i održavanja WEB aplikacija, koje su danas najzastupljenije na tržištu. Navedeno važi samo u slučaju kada se koristi template.



5. Nakon klika na „Finish“, trebalo bi da dobijete izgled okruženja sličan sledećem...



Na slici se mogu uočiti:

- **desno** – *TextEditor* sa generisanim kodom (to je za nas uradio kreacioni template) iznad kog se nalazi *linija sa aktivnim klasama* (trenutno samo Main klasa).
- **levo** – *Project* kartica u kojoj se može videti struktura projekta (hijerarhija njegovih fajlova i foldera).

Na njenom vrhu nalazi se folder (root folder) koji nosi naziv projekta koji smo definisali prilikom kreiranja projekta, a sa njim, u istom nivou se nalazi još i kolekcija biblioteka (3rd party biblioteke – kod specifične namene koji je neko drugi već razvio, a kojim se koristimo kako ne bi mi morali da radimo na tome).

U hijerarhiji root foldera mogu se primetiti i folderi „idea“ (sadrži konfiguracione fajlove samog okruženja) i „src“ (sadrži sav kod i samim tim i nama najbitniji).

U hijerarhiji foldera „src“ može se primetiti paket sa naziv koji smo ranije definisali i u njemu klasu koja, zbog upotrebe template-a, nosi naziv „Main“ i u sebi sadrži generisan kod sa „main()“ metodom. U slučaju da upotreba template-a nije odabrana u „src“ paketu bi se nalazio samo .iml fajl, a ostatak (pakete i klase bi ručno pravili).

- **gore** – iznad Project kartice i linije aktivnih klasa nalazi se *linija relativne putanje dela koda u Text editoru u kojem se nalazi kursor* (relativna putanja, za razliku od apsolutne, sadrži nazive samo onog dela hijerarhije koji se nalazi u oviru projekta, a ne svih, unutar particije, pa bi prema tome relativna putanja bila adresa datog entiteta u projektu – od root foldera projekta).

Napomena:

Praksa je izrodila određena pravila i konvencije prilikom imenovanja različitih delova koda.

Paketi se pišu malim slovima i upotrebom tačke – „.“ mogu se dodatno granati na potpakete.

Klase (kao i njeni posebni oblici **interfejsi** i **Enum** tipovi) se pišu velikim početnim slovom i njihova definicija (sav njihov sadržaj) nalazi se u okviru vitičastih zagrada – {}, koje slede nakon imena klase, tj. nakon navodjenja imena klase/interfejsa koju proširuju/implementiraju.

Promenljive se pišu malim početnim slovom, osim u slučaju kada se radi o *konstantama* (promenljive sa fiksnim vrednostima) kada se koriste sva velika slova.

Metodi se pišu malim početnim slovom, pri čemu se na kraju njihovog imena, u obliku zagradama – (), piše lista parametara (argumenti koje metod može da primi radi implementacije određene logike), iza kojih sledi eventualna lista izuzetaka i njegova definicija (definicija se piše u okviru vitičastih zagrada – {}).

Ukoliko je potrebno definisati ime nekog od navedenih delova koda koje će imati više reči, imajući u vidu da korišćenje razmaka (space) – ' ' nije dozvoljeno, koriste se *kamilja* ili *zmijska konvencija*.

Kamilja konvencija podrazumeva pisanje prve reči imena u skladu sa prethodno navedenim pravilima, dok se svaka naredna piše početnim velikim slovom.

Zmijska konvencija podrazumeva pisanje prve reči imena u skladu sa prethodno navedenim pravilima, dok se svaka naredna piše nakon znaka za 'donju crtu' – „_“. Iako se može u potpunosti samo ona koristiti, u praksi se najčešće koristi samo kod konstanti.

ImeKlase{def}

imePromenljive

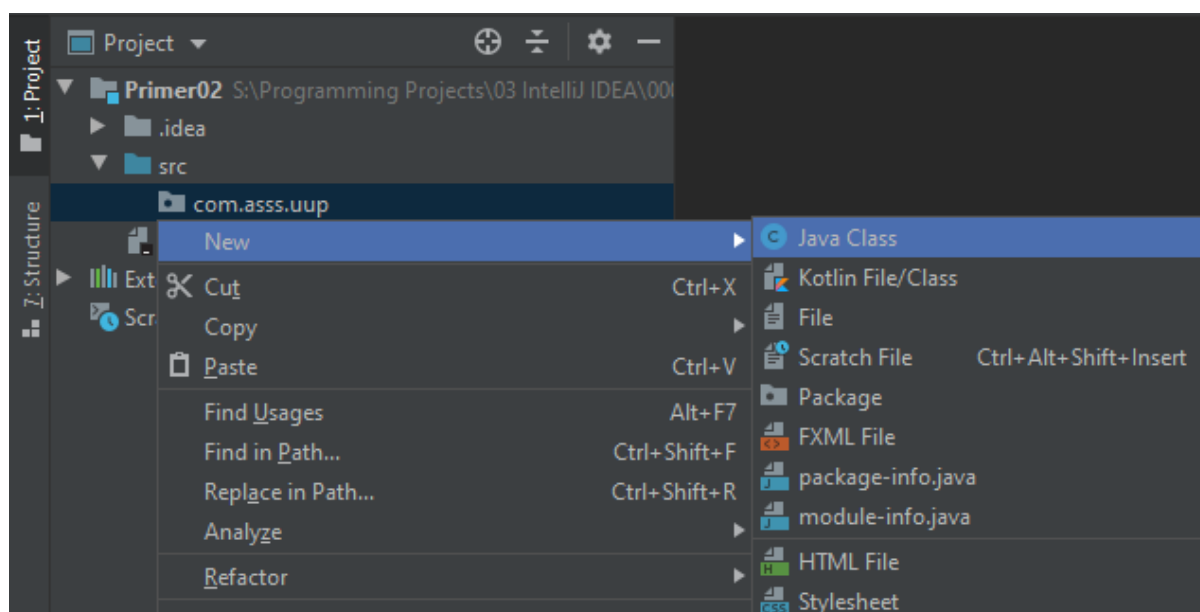
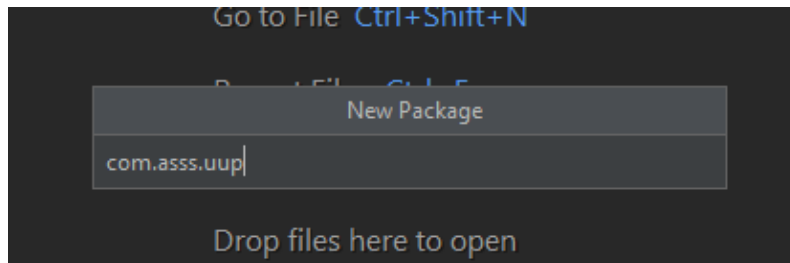
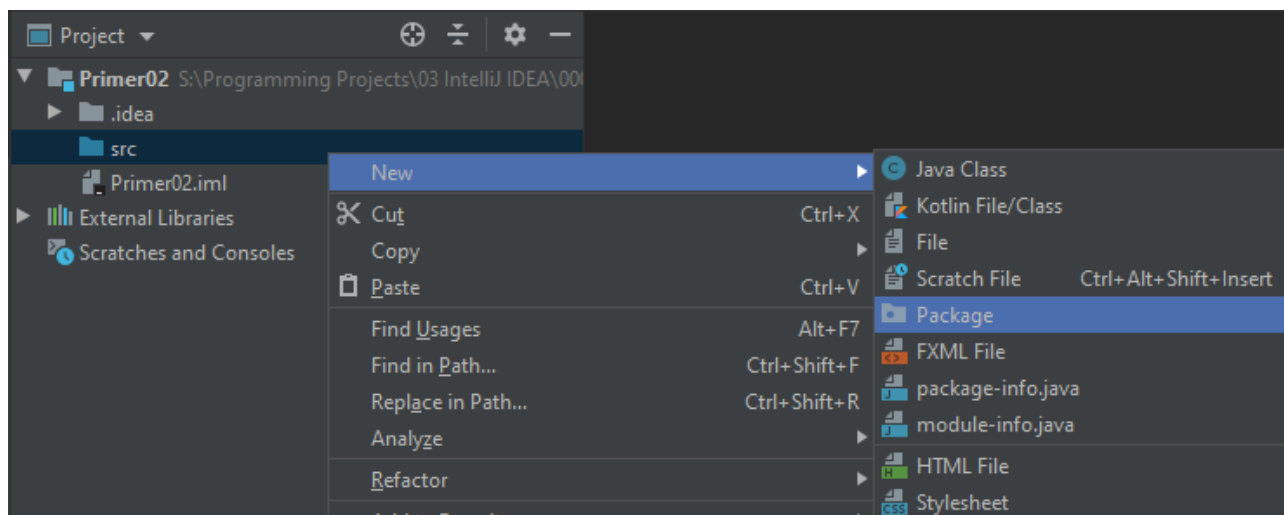
IME_KONSTANTE

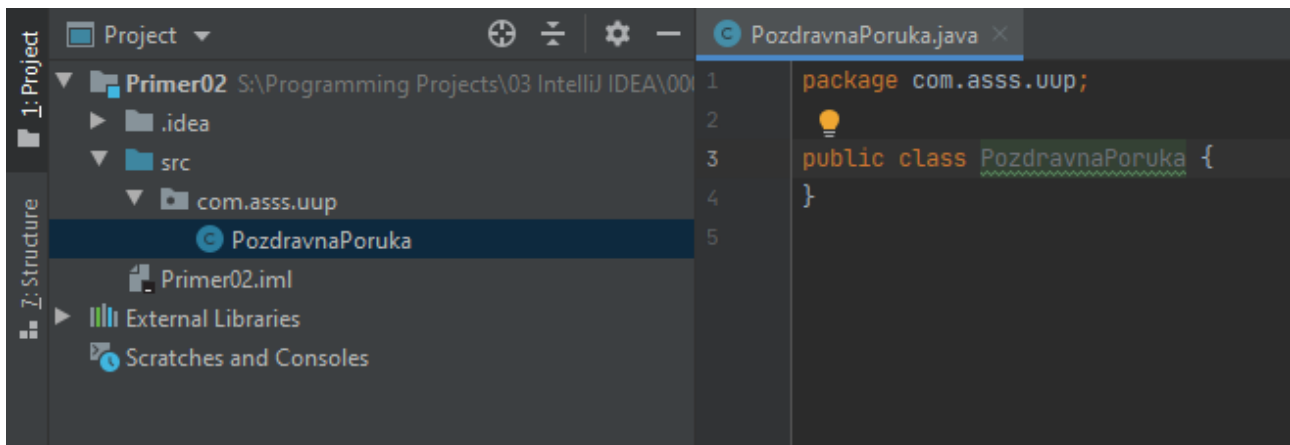
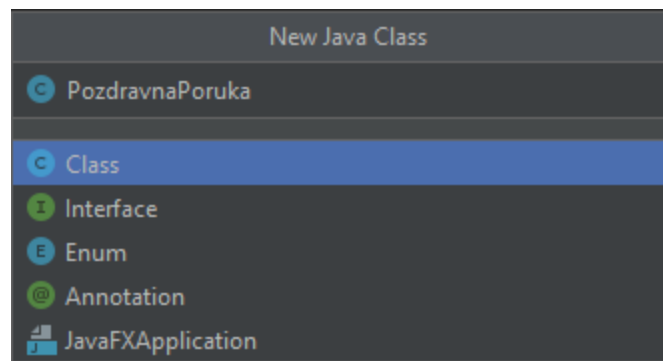
imeMetoda(args){def}

Primer 2 – String projekat (bez upotrebe Command Line App template-a)

1. U slučaju da prilikom kreiranja projekta nije odabrana upotreba CLA template-a, kao i u slučaju potrebe proširenja projekta, moguće je napraviti koliko god je potrebno paketa (i potpaketa), kao i klasa.

Potrebno je voditi računa da svaka klasa koja se napravi bude deo nekog konkretnog paketa (ko ima iskustava sa Eclipse IDE-om verovatno je upoznat sa pojmom i namenom „default“ paketa i taj pristup treba izbegavati).



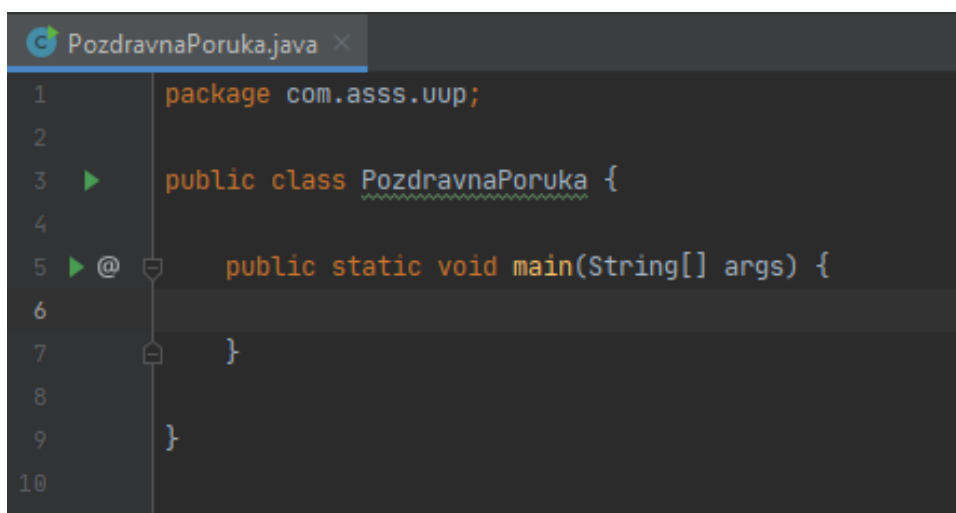


2. Pošto se napravi klasa, prelazi se na pisanje koda.

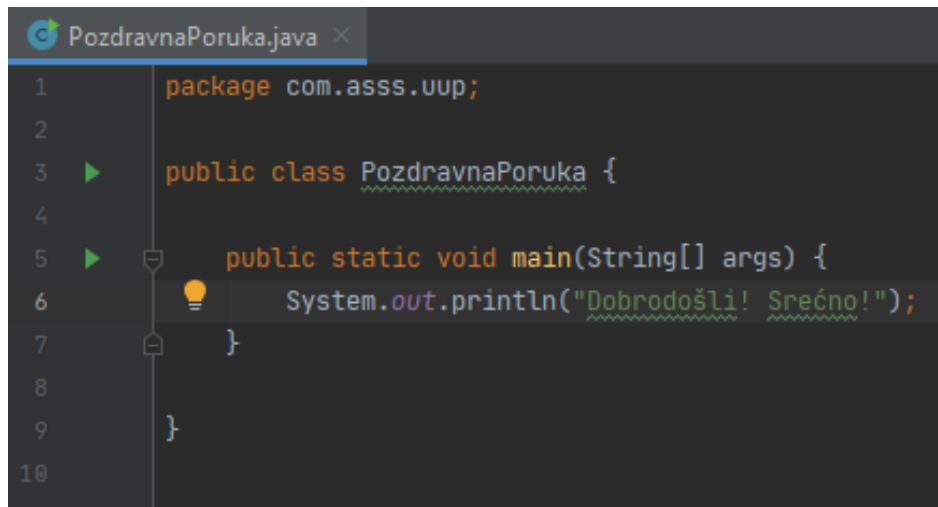
SAV KOD KOJI SE PIŠE MORA DA BUDE U OKVIRU DEFINICIJE KLASE!

U našem primeru, a kako bi došli do onog nivoa razvoja koji je obezbeđen upotrebom CLA template-a i kako bi se omogućilo izvršenje koda (biće objašnjeno i zašto), počinjemo sa definisanjem metoda „main()“.

Na ovom nivou potrebno je da znate da ni jedan projekat koji budemo pravili ne možete izvršiti bez upotrebe metoda main().



3. Nakon deklaracije metoda (prikazanoj na prethodnoj slici), prelazi se na definiciju metoda, tj. pisanje koda koji on treba da izvrši prilikom poziva.



```
1 package com.asss.uup;
2
3 public class PozdravnaPoruka {
4
5     public static void main(String[] args) {
6         System.out.println("Dobrodošli! Srećno!");
7     }
8
9 }
10
```

4. Pošto se potrebni kod ispiše i ukoliko nema nikakvih (sintaksnih) grešaka isti je moguće pokrenuti i izvršiti.

Imajući u vidu da se konzolne aplikacije (kroz koje ćete učiti programiranje tokom prvog semestra) izvršavaju shodno definiji main() metoda (u njemu počinje i u njemu se završava kompletan tok), pokretanje koda je moguće izvršiti kako nad njim direktno, tako i nad klasom u kojoj je definisan – glavna klasa.

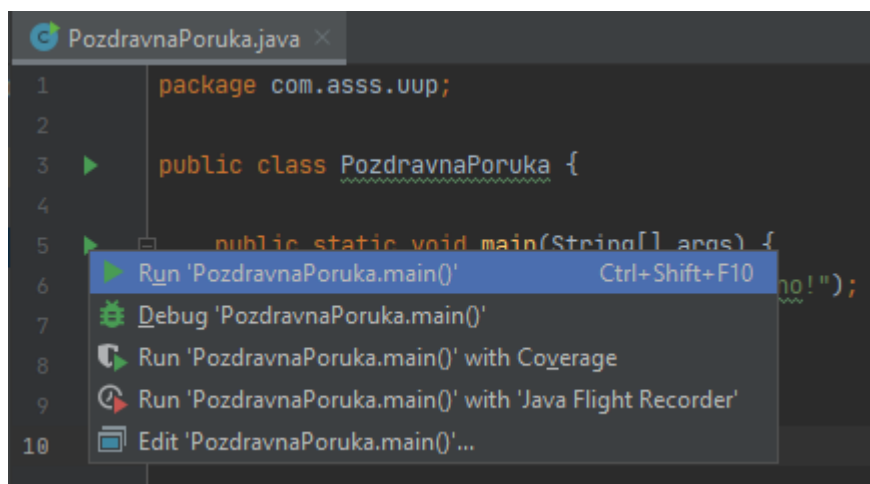
Napomena:

Kasnije, tokom rada, videćete da se projekat sastoji od više paketa koji u sebi imaju druge pakete – potpakete i da svi oni imaju klase sa svojim definicijama. Ono što je bitno da razumete na ovom nivou je to da je, kako je navedeno, izvršenje projekta isključivo definisano u okviru definicije main() metoda, a on kao takav, se piše u glavnoj (main) klasi, koju mi odredimo kao takvu.

Iako samo okruženje (IDE) može dozvoliti pisanje više main() metoda (samim tim i postojanje više glavnih klasa) to nije dobra praksa. Svi drugi metodi, kako u glavnoj, tako i u pomoćnim klasama (sve klase osim one u kojoj je definisan main() metod) su pomoćni metodi, tj. imaju zadatak da svojim implementacijama i shodno redosledu izvršavanja u okviru main() metoda omoguće adekvatno izvršavanje aplikacije.

Pokretanje projekta se može izvršiti klikom na zeleni trouglič, u traci sa numeracijom linija (leva strana Text Editor), bilo da je reč o onom koji se vezuje za samu (glavnu) klasu, bilo da se radi o onom koji je vezan sa main() metod, odabirom komande „Run ’ImeGlavneKlase.main()’“.

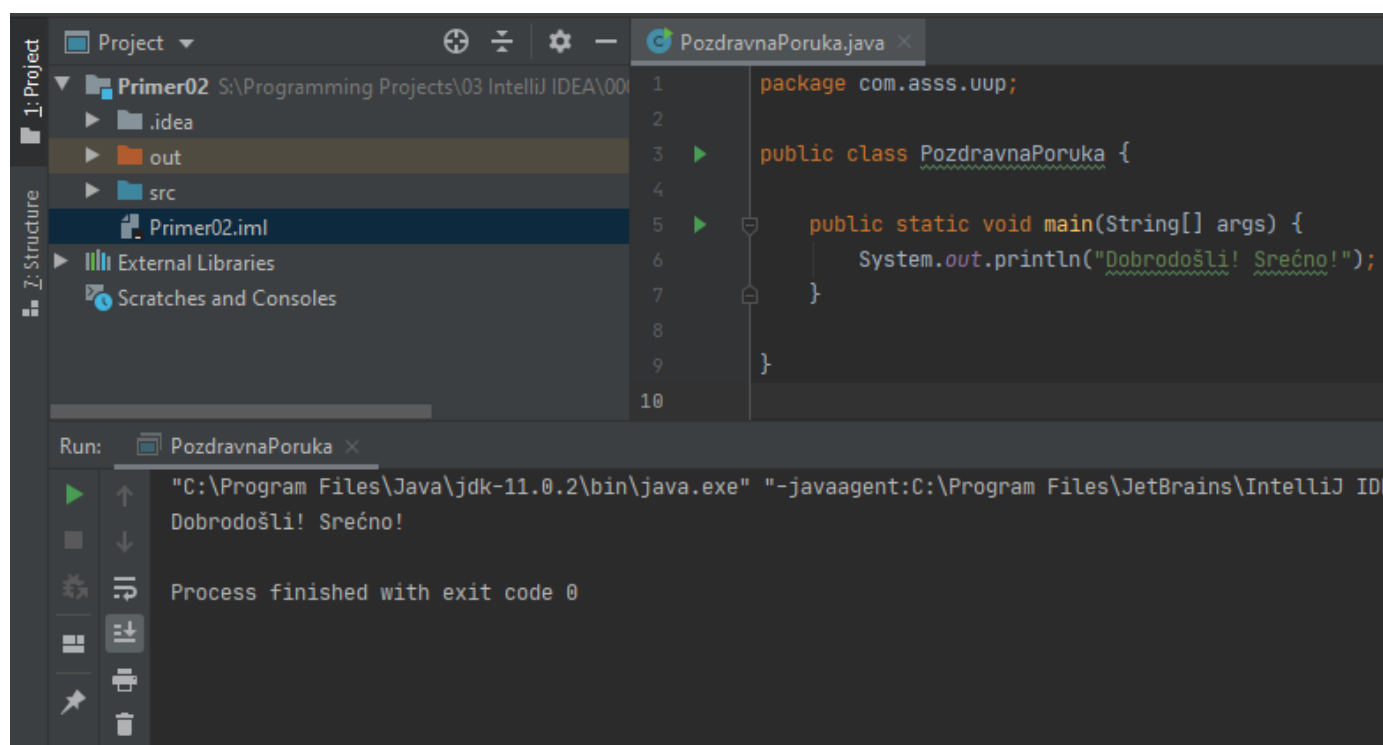
Pokretanje se takođe može izvršiti kombinacijom tastera ’Ctrl+Shift+F10’, klikom na zeleni trouglič u traci sa alatima (gornji desni ugao prozora IDE-a – treba voditi računa da je odabran željeni projekat), pomoću kartice prozora ’Run’ i odabirom prve opcije.



5. Nakon izvršenja programa u samom Text Editoru neće biti promena, dok će se u Project kartici (hijerarhija projekta) napraviti i „out“ folder koji sadrži kompajlirani kod – Java Bytecode.

Naravno, kao posledica izvršenja projekta prikazaće se i **konzola** (Run kartica) sa **rezultatom izvršenja koda**.

U konkretnom slučaju može se videti izveštaj sa JVM (prva i poslednja linija) i sam rezultat – pozdravna poruka koja je definisana u main() metodu.



Primer 3 – Projekat sa (celobrojnim) numeričkim vrednostima i objašnjenje primene različitih vrsta komentara (+ refactoring)

1. Pošto se, na već pokazan način, upotrebom CLA template-a, napravi novi projekat, kao što je viđeno, dolazi do generisanja *Main* klase i *main()* metoda u njoj.

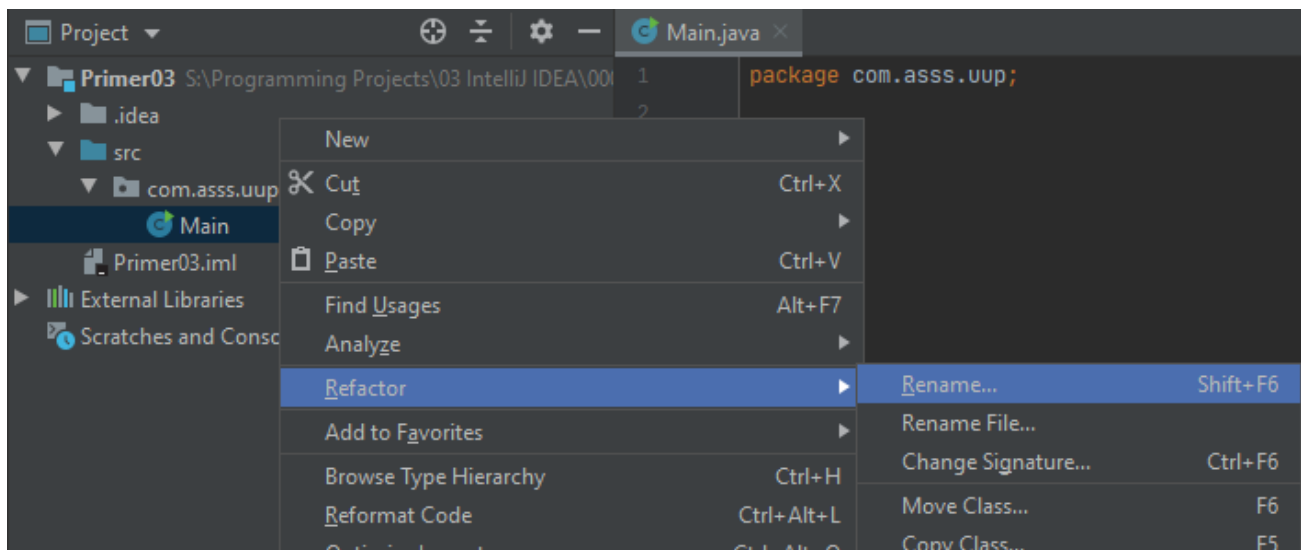
ImeProjekta > src > ime.paketa > Main

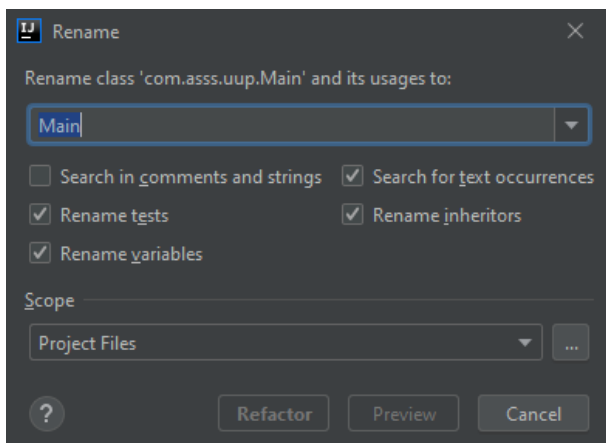
Generisanom metodu, zbog JVM-a i samog izvršenja programa, ne možemo menjati naziv, ali to nas ne sprečava da promenimo naziv klasi, čije ovako generisano ime možda nije najadekvatnije sa aspekta čitljivosti koda.

Treba imati u vidu da promena imena određenih delova koda nije prosta radnja kao kod promene imena nekog fajla i foldera (izvršiti uobičajno brisanje starog i pisanje novog imena), već je isto potrebno izvršiti kroz refaktorisanje (*Refactor*), čime dolazi do promene tog imena gde god je ono pozvano u celokupnom projektu.

Refactor nad određenim imenom (ispravno identifikatorom) može se izvršiti bilo u Project kartici, bilo u samom kodu, tako što se nad njim klikne desnim klikom, a potom odabere opcija *Refactor > Rename*. Isto se dobija kombinacijom tastera *Shift + F6*.

U zavisnosti od mesta gde se refactor izvršava može se dobiti 'popup' prozor sa opcijama ili samo selektovani identifikator koji treba promeniti (okvir oko imena).





```
public class Main {
    public static void main(String[] args) {
    }
}
```

2. Osnovna namena *komentara* je da pomognu u razvoju, a kasnije, i razumevanju napisanog koda.

Pored navedenog, njihova masovna primena se ogleda i u izolaciji koda. Naime, imajući u vidu da komentar ne predstavlja izvršivi deo klase (kompajler ga ignoriše), kao i to da kod gotovo nikada ne može da se napiše optimalno već da trpi određene izmene tokom faza razvoja i održavanja, iskustvo je pokazalo da je dobra praksa da u datom trenutku nepotrebne delovi koda ne treba brisati već jednostavno zakomentarisati. Tako je u svakom trenutku moguće vratiti se na neku ideju, implementirano rešenje i slično.

U radu sa komentarima treba razdvojiti bitno od nebitnog, odnosno pisati komentare konkretno i na mestima gde je to zaista potrebno (ne zatrpavati kod). Osnovna podela komentara i njihovo kratko pojašnjenje se nalazi na priloženoj slici, a više o tome možete naći u materijalu sa predavanja.

```
// jednolinijski komentar - obuhvata sve od mesta gde se postave dve uzastopne kose crte - "/" do kraja te linije

/*
višelinijski komentar - počinje kombinacijom karaktera "/" i "/*", a završava se kombinacijom karaktera "*/" i "/"
obuhvata sve što se nalazi između navedenih kombinacija karaktera
*/

/**
* dokumentacioni komentar - počinje kombinacijom karaktera "/" i "/*", a završava se kombinacijom karaktera "*/" i "/"
* obuhvata sve što se nalazi između navedenih kombinacija karaktera
* može i ne mora imati karakter "*" na početku svake linije
* razlikuje se po boji i nameni od prethodne dve vrste komentara
* koristi si prilikom razvoja softvera kako bi se omogućilo generisanje dokumentacije
* (Pri razvoju softvera dobra praksa je dati opis određenih celina u kodu i za to se koriste
* komentari. U slučaju upotrebe prve dve vrste opisu se može pristupiti samo ukoliko se
* ostvari uvid u sam kod, dok dokumentacioni komentar omogućuje generisanje HTML stranica
* na osnovu sadržaja opisa i one se stručno nazivaju dokumentacija.)
*/
```


3. Imajući u vidu da se upotreba aplikacija/programa zasniva na izvršenju određenih njihovih *metoda*, kao i to da se rad (izvršenje) metoda zasniva na izvršenju određenih operacija nad nekim *podacima*, potrebno je, pre svega, te podatke definisati, a potom implementirati logiku njihove obrade.

Kao što ste videli na predavanjima, podaci mogu biti *prostog* (ima ih osam) i *složenog tipa* (klase - obrađivaćemo ih u potpunosti kasnije tokom semestra), a da bi se mogli koristiti moraju imati *tip*, *identifikator* (naziv, ime) i *vrednost* (direktno zavisna od tipa – numerička, slovna, boolean, određenog opsega...).

U ovom primeru radićemo sa numeričkim celobrojnim tipom “**int**”.

```
// deklaracija promenljive "a" tipa "int"
int a;

// inicijalizacija promenljive "a"
a = 12;

// definisanje (deklaracija + inicijalizacija) promenljive "b"
int b = 7;

// deklarisanje više promenljivih istovremeno (moguće jedino ako su istog tipa)
int zbir, razlika, proizvod, količnik, ostatak;

// kodiranje matematičkih operacija shodno postavci zadatka
// postupak smeštanja vrednosti izračunavanja u promenljive u kojima treba čuvati rezultat naziva DODELA VREDNOSTI
zbir = a + b;
razlika = a - b;
proizvod = a * b;
količnik = a / b;
ostatak = a % b;

// prikaz rezultata na ekranu (u konzoli) - identično prethodnom primeru
System.out.println("Za potrebe izračunavanja koristimo sledeće operande a:" + a + " i " + " b:" + b + ".");

// prethodni zapis se može napisati i ovako...
System.out.print("Za potrebe izračunavanja koristimo sledeće operande");
System.out.print(" a:" + a);
System.out.print(" i");
System.out.print(" b:" + b);
System.out.println(".");

// prikazuje prazan red
System.out.println();
```

4. Da bi se primer doveo do kraja i dobio prikaz koji sledi potrebno je napisati još deo koda koji će omogućiti (“štampati”) prikaz u konzoli (uradite sami).


```
Za potrebe izračunavanja koristimo sledeće operande a:12 i b:7.  
  
Rezultat operacije sabiranja nad operandima a i b je 19.  
Rezultat operacije oduzimanja nad operandima a i b je 5.  
Rezultat operacije množenja nad operandima a i b je 84.  
Rezultat operacije deljenja nad operandima a i b je 1.  
Rezultat operacije ostatka pri deljenju nad operandima a i b je 5.
```

Primer 4 – Upotreba klasa iz Javine biblioteke (klasa `java.lang.Math`)

1. Kako je navedeno na predavanjima, instalacijom JRE-a, odnosno JDK-a koji sadrži i JRE, dobija se i velika kolekcija Javinih biblioteka. Te biblioteke su sastavljene od klasa različite namene i njih je moguće koristiti u kodu kako ne bi gubili vreme na pisanje potrebnih algoritama (kao u prethodnom primeru).

Sve klase biblioteke „`java.lang`“ su prema podrazumevanim vrednostima uvezene (kasnije tokom predavanja će biti reči o uvozu – `import`-u klasa) u svaki projekat koji se napravi i njihovi elementi (polja i metodi) se mogu pozivati upotrebom kvalifikovanog imena, dok s druge strane, da bi koristili sve ostale Javine klase, iz drugih biblioteka iste moramo imenovati punim kvalifikovanim imenom.

Nakon što se klasa uveze u projekat (našu klasu) njeni elementi se mogu pozivati nekvalifikovanim imenima.

`imePaketa.imeKlase.imeElementa` – puno kvalifikovano ime

`imeKlase.imeElementa` – kvalifikovano ime

`imeElementa` – nekvalifikovano ime

U ovom primeru radićemo sa klasom „`Math`“ koja pripada „`java.lang`“ paketu, gde ćemo prilikom upotrebe njenih metoda koristiti kvalifikovano ime.

Klasa „`Math`“ pruža punu podršku za sve matematičke operacije nad različitim tipovima numeričkih vrednosti i kao takva značajno olakšava razvoj softvera.

2. Poučeni iskustvom iz prethodnog primera, da prilikom rada sa celobrojnim tipovima podataka može doći do gubitka vrednosti (operacija deljenja ne daje ostatak, niti njegov decimalni iznos), u ovom primeru ćemo raditi i sa numeričkim tipom „`double`“ čije su vrednosti brojevi u pokretnom zarezu – decimalni brojevi.

```
/*
Napraviti program koji ce sadržati dve promenljive i obezbediti prikaz njihovog zbira, razlike, proizvoda,
količnika, proizvoda, ostatka pri deljenju, kao i kvadratni stepen prve i kvadratni koren druge promenljive.
Za sve navedene operacije koristiti isključivo klasu Javinu klasu Math.
*/
public class Main {

    public static void main(String[] args) {

        // definisanje promenljivih za operande
        int a = 12, b = 7;

        // deklarisanje promenljivih za rezultate
        double zbir, razlika, proizvod, količnik, ostatak, kvadrat, koren;

        // izdračunavanje upotrebom klase Math
        zbir = Math.addExact(a,b);
        razlika = Math.subtractExact(a,b);
        proizvod = Math.multiplyExact(a, b);
        količnik = Math.floorDiv(a, b);
        ostatak = Math.floorMod(a, b);
        kvadrat = Math.pow(a, 2);
        koren = Math.sqrt(b);
    }
}
```

3. Nakon koda prikazanog na slici, kao kod prethodnog primera, potrebno je ispisati deo koda koji će obezbediti prikaz u konzoli shodno izlazu na narednoj slici.

```
Za potrebe izračunavanja koristimo sledeće operande a:12 i b:7.

Rezultat operacije sabiranja nad operandima a i b je 19.0.
Rezultat operacije oduzimanja nad operandima a i b je 5.0.
Rezultat operacije množenja nad operandima a i b je 84.0.
Rezultat operacije deljenja nad operandima a i b je 1.0.
Rezultat operacije ostatka pri deljenju nad operandima a i b je 5.0.
Kvadratni stepen operanda a 144.0.
Kvadratni koren operanda b 2.6457513110645907.
```

Primer 5 – Upotreba klasa iz Javine biblioteke (klasa java.util.Scanner)

1. Podaci koje treba da obezbedimo programu mogu biti napisani direkno u kodu („hardcoded vrednosti“ – veoma retko, u praksi se taj slučaj svodi uglavnom na konstante) ili/i učitani u program.
Najčešći oblici učitavanja podataka svode se na baze podataka, fajlove/dokumente i korisnički unos – unos sa tastature.

Najjednostavniji način da se obezbedi korisnički unos tokom izvršenja (konzolne) aplikacije predstavlja upotreba Javine klase „Scanner“ (java.util.Scanner).

Kao što se može videti iz punog kvalifikovanog imena klase, ona pripada paketu „java.util“ što znači da ili je moramo uvesti u program (iskaz „import“) ili pozivati njene članove stalno navodeći njihovo puno kvalifikovano ime (što nije dobra praksa).

Imajući u vidu da je za razumevanje klase potrebno osnovno znanje OOP-a s kojim ćete se sresti u drugom delu semestra, potrebno je „uzeti zdravo za gotovo“ princip upotrebe elemenata klase Scanner, jer će se ista koristiti u međuvremenu.

2. Predstojećim primerom obuhvatićemo prethodno stečena znanja upotrebe klasa String, Math, sistemskog izlaza, u kombinaciji sa korisničkim unosom koristeći klasu Scanner.

```
/*
Napraviti program koji će upotrebom klase Scanner omogućiti unos i izračunavanje sledećih podataka:
a) Unos imena i prezimena, godine starosti, zatim broja indeksa i na kraju nastavne grupe;
b) Izračunavanje vrednosti površine P i hipotenuze c pravouglog trougla na osnovu unetih vrednosti kateta a i b;
c) Izračunavanje vrednosti površine P i obima O pravougaonika na osnovu unetih vrednosti stranica a i b.
Obezbediti da unos podataka sa tastature bude tekstualno praćen u konzoli, kao i prikaz krajnjih rezultata.
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        // deklarisanje promenljivih u kojima će se čuvati podaci definisani u tački a)
        String ime, prezime, brojIndeksa, nastavnaGrupa;
        int godine;

        // deklarisanje promenljivih za katete/stranice
        int aT, bT, aP, bP;

        // deklarisanje promenljivih za površinu P (trougla, pravougaonika), te za hipotenuzu c i obim O
        double c, površinaTrougla, površinaPravougaonika, obimPravougaonika;

        // pravljenje objekta tipa Scanner
        Scanner unosSaTastature = new Scanner(System.in);
```

3. Pošto se deklarishu potrebne promenljive, potrebno je, shodno zahtevima, napisati logiku kojom će se zadatak izvršavati.

Napomena:

U poslednjoj liniji koda prikazanog na slici se može ostvariti uvid u to kako se pravi objekat klase Scanner, dok će se na nekim od narednih slika može videti upotreba njegovih određenih metoda (nextLine(), nextInt(), close()) kojima se omogućuje unos podataka sa tastature, odnosno zatvaranje Scanner objekta kao resursa.

4. Narednom slikom prikazan je kod koji zadovoljava zahteve definisane u tački a).

Treba obratiti pažnju na upotrebu metoda Scanner klase.

Metod *nextLine()* u vidu String objekta (znakovnog niza) učitava sve što je uneto na tastaturi i smešta u promenljivu kojoj treba da dodeli vrednost. Prilikom rada sa višerečnim tekstualnim unosom treba voditi računa da se greškom ne pozove metod *next()* koji učitava samo prvu reč.

Metod *nextInt()* uneti sadržaj učitava kao celobrojni broj. Kod njega treba obratiti pažnju da se ne unese nikakva drugačija vrednost (npr. ne bi mogao da se iskoristi za učitavanje broja indeksa jer broj indeksa zadrži kosu crtu). U slučaju pokušaja učitavanja tipa podatka koji nije adekvatan Scanner objektu, dolazi do greške i prekida izvršenja programa (metod *nextLine()* će bez problema učitati numeričke vrednosti ali će ih posmatrati kao tekst – String objekat).

U prikazanom kodu postoji jedna stvar koja može privući pažnju. Naime, kod unosa godina, pozvan je metod *nextInt()* koji treba da učitava vrednost unetu na tastaturi, ali odmah potom je i pozvan metod *nextLine()* koji „ne učitava ništa“. To je iz razloga što da bi se potvrdio unos broja (godina) potrebno je stisnuti taster „Enter“, kao i kod svih drugih vrednosti. Međutim, kada se poziva poziva metod *nextInt()*, kom sledi metod *nextLine()*, dolazi do pojave da „Enter“ koji korisnik unese za potvrdu broja, metod *nextLine()* pokupi kao svoju vrednost i time onemogućiti unos tekstualnog podatka koji je trebalo metod *nextLine()* da prihvati.

Da biste shvatili o čemu se tačno radi zakomentarišite poziv metoda *nextLine()* u delu koda gde se radi sa godinama i prilikom izvršenja koda primetićete da, pošto unesete godine, program jednostavno „preskoči“ unos broja indeksa (zapravo je stavio praznu vrednost jer je pokupio „Enter“) i prebacio tok na unos nastavne grupe.

```
System.out.println("a");           // ispis tačke za koju se unose podaci

System.out.print("Ime: ");          // ispis teksta pre unosa podatka sa tastature
ime = unosSaTastature.nextLine();  // učitava sve što se unese sa tastature kao String

System.out.print("Prezime: ");      // ispis teksta pre unosa podatka sa tastature
prezime = unosSaTastature.nextLine(); // učitava sve što se unese sa tastature kao String

System.out.print("Godine: ");       // ispis teksta pre unosa podatka sa tastature
godine = unosSaTastature.nextInt(); // učitava vrednost unetu sa tastature kao int (broj)
unosSaTastature.nextLine();         // ova linija koda kupi novi red koji nastaje izvršenjem prethodne

System.out.print("Broj indeksa: "); // ispis teksta pre unosa podatka sa tastature
brojIndeksa = unosSaTastature.nextLine(); // učitava sve što se unese sa tastature kao String

System.out.print("Nastavna grupa: "); // ispis teksta pre unosa podatka sa tastature
nastavnaGrupa = unosSaTastature.nextLine(); // učitava sve što se unese sa tastature kao String
```

5. Narednom slikom prikazan je kod koji odgovara na zahteve tačke b) i c) po pitanju unosa traženih podataka i zatvara Scanner objekat, kao resurs.

```
System.out.println();           // ispis teksta pre unosa podatka sa tastature
System.out.println("b");       // ispis tačke za koju se unose podaci

System.out.print("Kateta a = "); // ispis teksta pre unosa podatka sa tastature
aT = unosSaTastature.nextInt(); // učitava vrednost unetu sa tastature kao int (broj)

System.out.print("Kateta b = "); // ispis teksta pre unosa podatka sa tastature
bT = unosSaTastature.nextInt(); // učitava vrednost unetu sa tastature kao int (broj)

System.out.println();           // ispis teksta pre unosa podatka sa tastature
System.out.println("c");       // ispis tačke za koju se unose podaci

System.out.print("Stranica a = "); // ispis teksta pre unosa podatka sa tastature
aP = unosSaTastature.nextInt(); // učitava vrednost unetu sa tastature kao int (broj)

System.out.print("Stranica b = "); // ispis teksta pre unosa podatka sa tastature
bP = unosSaTastature.nextInt(); // učitava vrednost unetu sa tastature kao int (broj)

// pošto se završi sa upotrebom određenog resursa (baza, fajl, unos...) potrebno je isti zatvoriti
// u može doći do problema nad njim i nad upravljanjem memorijom (resource leak)
unosSaTastature.close();
```

```
a)
Ime: Nenad
Prezime: Stanković
Godine: 28
Broj indeksa: 12/7
Nastavna grupa: 6P

b)
Kateta a = 7
Kateta b = 12

c)
Stranica a = 9
Stranica b = 27
```

Ukoliko je sve urađeno kako treba tokom izvršenja programa i unošenja vrednosti sa tastature trebalo bi da dobijete sličan ispis u konzoli.

Zelenim slovima je prikazan unos sa tastature, dok je belim „štampano“ sve navedeno kao argument metoda print(), odnosno println().

6. Na kraju preostaje da se implementira logika izračunavanja traženih vredosti u tačkama b) i c) i prikaz svih prikupljenih (unetih na tastaturi) i izračunatih vrednosti u konzoli.

Taj deo ostaje vama da uradite samostalno, a načelan prikaz rezultata je prikazan na sledećoj slici...

```
a)
Ime: Nenad
Prezime: Stanković
Godine: 28
Broj indeksa: 12/7
Nastavna grupa: GP

b)
Kateta a = 7
Kateta b = 12

c)
Stranica a = 9
Stranica b = 27

Student Nenad Stanković (28) sa brojem indeksa 12/7 je raspoređen u grupu GP.
Rezultati:
Trougao - katete: 7 i 12, hipotenuza: 13.892443989449804, P = 42.0
Pravougaonik - stranice: 9 i 27, P = 243.0, O = 72.0
```

Ukoliko bude nekih pitanja, nedoumica ili nejasnoća pitajte na sledećim vežbama ili pošaljite mejlom, jer zbog tempa koji nam je nametnut možda nećemo stići da sve obradimo na vežbama, tj. licem u lice.

Ono što je bitno je da probate! Niko nije naučio, a da nije pogrešio. Treba da grešite! Da grešite i da učite! Istražujte, budite uporni i ako bude potrebe da pitate, pitajte, ali očekujem konkretno pitanje i konkretne predloge mogućih rešenja.

Bavite se problemom, jer se ovaj posao tako radi!

Još jednom, želim vam mnogo sreće i uspeha, kako tokom školovanja, da steknete osnovno znanje, tako i nakon njega, da počnete da ga primenjujete i proširujete!