1. domaća zadaća

Prije no što krenete dalje, pročitajte uputu na kraju dokumenta.

1. Početno podešavanje

Ako još niste s interneta preuzeli posljednju inačicu Eclipse-a, učinite to. Starije inačice vjerojatno neće znati korektno raditi s Javom 13.

Pokrenite *Eclipse* i odaberite *workspace* direktorij (ako ćete koristiti neki drugi, a ne onaj koji ste već napravili na predavanjima). Kad se *Eclipse* pokrene, na disku će napraviti taj direktorij, i u njemu (vjerojatno) poddirektorij `RemoteSystemsTempFiles` kao i skriveni poddirektorij `.metadata`.

Ako ste radili novi workspace, provjerite da za pohranu tekstovnih datoteka koristi UTF-8.

Napravite kroz Eclipse novi Java projekt imena `hw01-0000000000` (zamijenite nule Vašim JMBAGom). Time ste spremni za rješavanje zadataka koji slijede.

Kada radite prvi program, napravite mišem desni klik u *Project Exploreru* na naziv direktorija u koji treba smjestiti izvorni kod (`*.java` datoteke) – u našem slučaju na `src`, pa u iskočnom izborniku odaberite `New → Class`. U prozoru koji se otvori, pri vrhu imate mjesto gdje možete upisati naziv paketa u koji treba smjestiti razred/program a malo ispod toga i mjesto gdje možete upisati naziv razreda/programa. Alternativno, iz iskočnog izbornika možete odabrati `New → Package`, pa zatim desnim klikom na sam paket u njemu stvarati razrede. Sve zadatke iz ove zadaće smještat ćete u paket `hr.fer.zemris.java.hw01`.

Prilikom pisanja koda, pridržavajte se sljedećih konvencija.

- Nazivi razreda/sučelja/enumova uvijek se pišu velikim početnim slovom. Ako su konkatenacija više riječi, svaka sljedeća opet započinje velikim početnim slovom. Npr. `Student`, `StudentMail`, `StudentCourseEnrolment`.
- Nazivi metoda uvijek se pišu malim početnim slovom. Ako su konkatenacija više riječi, svaka sljedeća opet započinje velikim početnim slovom. Npr. `print`, `printGrades`, `printStudentGrades`. Nikada nemojte kratiti imena (primjerice izbacivanjem suglasnika, što zna biti praksa u C kodu; dakle ne: `prnt`, `prntGrds` i slično). Također, riječi nikad ne spajajte podvlakama (također praksa iz C-a); znači ne: `print_student_grades`.
- Nazivi varijabli slijede istu konvenciju kao i metode: `dayOfMonth`, `monthOfYear`, itd.
- Varijable deklarirajte uvijek tamo gdje ih prvi puta trebate. Nemojte nikada deklarirati sve što Vam treba na početku metode. Varijable najčešće inicijalizirate odmah pri deklaraciji, Dakle, ne:
  ```
  int month;
  month = 7;
  ```
  već
  ```
  int month = 7;
  ```
  Također, ne:
  ```
  int month;
  month = calculateMonth();
  ```
  već:
  ```
  int month = calculateMonth();
  ```
- Doseg varijable treba biti što je manji (uži) moguć. Ovo je korektno:
  ```
  int i;
  ```

```
for(i = 0; i < 10; i++) {
 ...
}
```
ali ako nam taj `i` ne treba iza petlje `for`, onda je puno bolje:
```
for(int i = 0; i < 10; i++) {
 ...
}
```

- Nove varijable mogu se definirati i unutar blokova. Primjerice, ako nam treba varijabla koja postoji samo u tijelu naredbe `if`, tamo je deklarirajte. Npr.
```
if(month > 6) {
  double bonusFactor = 1.2;
  ... radi nešto s bonusFactor ...
}
```
- Obavezno korektno indentirajte kod!
- Obavezno pišite Javadoc kako smo objasnili na predavanju.
- Uočite kako smo u prethodnim primjerima radili s vitičastim zagradama. Vitičasta zagrada otvara se u nastavku (u istom retku, nakon jedne praznine) naredbe kojoj taj blok pripada (pogledajte gore primjer naredbi `if` i `for`), u sljedećim retcima uvučeno dolaze naredbe koje pripadaju tom bloku, i potom u novom retku vizualno poravnatno okomito prema gore s prvim slovom naredbe kojoj blok pripada dolazi zatvorena vitičasta zagrada (kod `if`-a je poravnata sa slovom `i`, kod `for`-a sa slovom `f`).
- Naredbe `if`, `for`, `do` i while uvijek pišite s pripadnim blokovima, čak ako je unutra i samo jedna naredba. Dakle, ne:
```
if(a > b) printSomething();
```
već:
```
if(a > b) {
  printSomething();
}
```

Zadatak 1.

U paketu `hr.fer.zemris.java.hw01` napravite program `Factorial`. Program se pokreće bez argumenata. Korisnik preko tipkovnice unosi cijele brojeve u rasponu od 3 do 20. Ako korisnik zada broj koji je izvan tog raspona ili ako nije broj, ispisati odgovarajuću poruku i nastaviti dalje s radom. Program ispisuje faktorijelu zadanog broja. Odaberite za izračun primitivni tip podataka uz koji ćete moći izračunati rezultat za ovaj raspon brojeva. Evo očekivanog primjera interakcije programa i korisnika, nakon što se program pokrene:

```
Unesite broj > štefica
'štefica' nije cijeli broj.
Unesite broj > 3
3! = 6
Unesite broj > 3.14
'3.14' nije cijeli broj.
Unesite broj > -4
'-4' nije broj u dozvoljenom rasponu.
Unesite broj > 4
4! = 24
Unesite broj > kraj
Doviđenja.
```

Metoda za sam izračun faktorijele ne smije ništa ispisivati na ekran. Rad programa prekida se kada korisnik unese "`kraj`". Korisnikovi unosi prikazani su crvenom bojom. Vaš bi program za identične ulaze iz primjera trebao generirati upravo prikazani ispis.

Izračun faktorijele ostvarite u zasebnoj pomoćnoj metodi (složenost nije bitna). Ako metoda primi vrijednost argumenta za koju ne može izračunati faktorijelu (jer faktorijela za dani broj ne postoji, ili pak nije prikaziva u broju bitova koji koristite za povrat vrijednosti pozivatelju), treba baciti iznimku `IllegalArgumentException`. Primijetite da se ograničenja na korisniku dozvoljen raspon brojeva općenito (pa tako i ovdje) razlikuju od ograničenja na raspon brojeva koji Vaša metoda zna izračunati. Razmislite gdje ćete u kodu kontrolirati što, i kako ćete organizirati program.

Zadatak 2.

U paketu `hr.fer.zemris.java.hw01` napravite program `Rectangle`. Program pita korisnika preko naredbenog retka da unese širinu pa visinu pravokutnika. Program korisniku ispisuje površinu i opseg pravokutnika. Ako program pri pokretanju dobije argumente preko naredbenog retka, onda ništa ne pita korisnika, već odmah računa i ispisuje površinu i opseg. Korisnik kao širinu i visinu može unijeti i decimalne brojeve. Primjerice, ako program pokrenemo ovako:

```
Rectangle 2 8
```

program će odmah ispisati sljedeće, i nakon toga prekinuti s radom.

```
Pravokutnik širine 2.0 i visine 8.0 ima površinu 16.0 te opseg 20.0.
```

Ako ga pokrenemo bez argumenata, očekivano je ponašanje ilustrirano ispisom u nastavku.

```
Unesite širinu > štefica
'štefica' se ne može protumačiti kao broj.
Unesite širinu > -2.1
Unijeli ste negativnu vrijednost.
Unesite širinu > 2
Unesite visinu > štefica
'štefica' se ne može protumačiti kao broj.
Unesite visinu > -2.1
Unijeli ste negativnu vrijednost.
Unesite visinu > 8
Pravokutnik širine 2.0 i visine 8.0 ima površinu 16.0 te opseg 20.0.
```

Prilikom pisanja programa nemojte sav kod nagurati u metodu `main`. Stvari koje se ponavljaju izolirajte u pomoćne metode pa ih pozivajte iz metode `main` koliko puta je potrebno.

Ako se programu preda broj argumenata koji je različit od 0 i različit od 2, program treba ispisati prikladnu poruku i prekinuti s radom.

Upozorenje: čitanje s tipkovnice riješite uporabom razreda `Scanner`, i to tako da čitate redak po redak (nemojte koristiti `hasNextDouble()` i `nextDouble()` jer ova posljednja koristi lokalizacijske postavke pa će nekad prihvaćati decimalnu točku, a nekad decimalni zarez). Umjesto toga čitajte kompletan redak po redak (pokazali smo na predavanju kako), iz pročitanog retka uklonite vodeće i prateće praznine ako postoje (razred `String` za to ima prikladnu metodu), te ono što je preostalo uporabom prikladne metode iz razreda omotača `Double` isparsirajte u `double`; ako se to ne da, ispišite prikladnu poruku kako je ilustrirano u primjeru. Ovo će osigurati da korisnik uvijek unosi decimalnu točku. Pri ispisu brojeva također se mora koristiti decimalna točka, što statička metoda `Double.toString(...)` poštuje.

Zadatak 3.

U paketu `hr.fer.zemris.java.hw01` napravite program `UniqueNumbers`. Program se pokreće bez argumenata. U programu napravite javnu pomoćnu strukturu `TreeNode` koja ima članske varijable `left` i `right`, te `value` koji je tipa `int` (ne smije imati *ništa* osim toga) – pogledajte u primjerima s predavanja kako smo u programu dodali pomoćnu strukturu `MyInteger`; struktura `TreeNode` samo ima par varijabli više. Napravite u programu pomoćne metode koje dodaju čvor u uređeno binarno stablo (lijevo manji, desno veći), samo ako čvor s vrijednosti koja se dodaje već ne postoji (u suprotnom, metoda ne radi ništa). Prisjetite se: alociranje objekta koji je takvog tipa radimo operatorom `new` koji nam vraća referencu na alocirani objekt. Evo primjera uporabe.

```
TreeNode glava = null;
glava = addNode(glava, 42);
glava = addNode(glava, 76);
glava = addNode(glava, 21);
glava = addNode(glava, 76);
glava = addNode(glava, 35);
```

Ako ste to dobro implementirali, stablo će imati četiri čvora i vrijedit će:

```
glava.value: 42
glava.left.value: 21
glava.left.right.value: 35
glava.right.value: 76
```

Metoda `add` dakako ništa ne ispisuje na zaslon. Napravite metodu koja vraća veličinu stabla. Evo primjera uporabe.

```
int velicina = treeSize(glava);
```

Pozovemo li je nakon prethodnog primjera, rezultat bi morao biti 4. Pozovemo li je prije prvog poziva `addNode`, rezultat bi morao biti 0.

Napravite metodu `containsValue` koja vraća nalazi li se traženi element u stablu. Primjerice, sljedeće bi nakon prethodnog primjera trebalo ispisati `da`.

```
if(containsValue(glava, 76)) {
  System.out.println("da");
}
```

Primijetite ponovno da sama metoda `containsValue` ništa ne ispisuje na zaslon.

Potom napravite glavni program tako da od korisnika s tipkovnice čita broj po broj i dodaje ih u stablo, samo ako tamo već ne postoje (a što se točno dogodilo, ispisuje na zaslon). Za rad sa stablom koristite samo prethodno napisane metode; nemojte pisati novu koja trči po stablu i gleda može li dodati element, pa ako ne može, ispisuje poruku na ekran, a inače dodaje element – to rješavate na drugom mjestu! Korisnik unos prekida utipkavanjem "`kraj`". Jednom kad je unos gotov, program treba ispisati brojeve najprije sortirano od manjeg prema većem, a potom u sljedećem retku od većeg prema manjem. Evo primjer očekivane interakcije.

```
Unesite broj > štefica
'štefica' nije cijeli broj.
Unesite broj > 3.14
'3.14' nije cijeli broj.
Unesite broj > 42
Dodano.
Unesite broj > 76
Dodano.
```

```
Unesite broj > 21
Dodano.
Unesite broj > 42
Broj već postoji. Preskačem.
Unesite broj > 35
Dodano.
Unesite broj > kraj
Ispis od najmanjeg: 21 35 42 76
Ispis od najvećeg: 76 42 35 21
```

Ako zbog bolje organizacije koda trebate pomoćne metode, slobodno ih napravite.

Napomena.

Prilikom rješavanje svih domaćih zadaća, pa tako i ove, zadatke morate rješavati samostalno. Smijete se konzultirati s drugim polaznicima vještine PRIJE no što krenete programirati (što treba riješiti, koja je ideja, kako bi se rješenje moglo oblikovati i slično). Jednom kad krenete programirati, dužni ste sami samostalno napisati rješenje. Ako nešto tada zapne, postoje konzultacije. Gledanje tuđeg rješenja domaće zadaće ili čak uporaba te krađa dijelova koda i ugradnja istoga u vlastitu domaću zadaću smatra se varanjem (bilo da ste nakon toga uhvaćeni ili ne – budite svjesni ako to radite, da to radite) i imat će posljedice. Proučite još jednom ostatak pravila iz uvodne (nulte) prezentacije. Ako ste ovakve ili slične zadatke već rješavali u okviru nekog drugog kolegija, ne smijete ovdje predati copy&paste tog koda – rješite zadatke ponovno samostalno.

U ovoj prvoj zadaći ne smijete koristiti biblioteke koje nismo obradili; posebice ne smijete koristiti *Java Collection Framework* te gotove implementacije kolekcija. Ako niste sigurni smijete li nešto koristiti, postavite pitanje u Ferku na *Pitanja i odgovori* (imate na stranici kolegija link gore na vrhu ekrana u drugoj traci). Pomoćne strukture ovdje koristimo doslovno kao strukture, emulirajući programski jezik C; ne smijete u njima pisati Vaše konstruktore, pomoćne metode i slično.

Pročitajte u knjizi dio poglavlja 2 (stranice 21-32) te poglavlje 4 (stranice 75 do 91). Također, prođite još jednom ako je potrebno kroz dokument koji smo koristili na predavanju.

Predaja domaće zadaće.

Eclipse projekt potrebno je zapakirati u arhivu imena `hw01-0000000000.zip` (zamijenite nule Vašim JMBAG-om). Evo postupka. Napravite desni klik mišem na naziv projekta u `Package Exploreru` ("ono" uz lijevi rub prozora). U dijalogu koji se otvori proširite rubriku `General`, pa odaberite `Archive File`. U dijalogu koji slijedi gornji dio će biti razdijeljen u lijevi i desni dio. U lijevom dijelu proširite Vaš projekt, i prikazat će se direktoriji koji su u njemu. Ostavite označen `src`, a uklonite kvačicu uz `.settings` i `bin`. U direktoriju samog projekta u desnom dijelu prozora trebaju biti označene datoteke `.classpath` i `.project`. Pri sredini dijaloga nalazi se "`To arhive file:`" gdje trebate upisati stazu i naziv arhive koju treba stvoriti. Pod opcijama pri dnu treba biti označeno da želite ZIP-arhivu. Jednom kad ste izgenerirali ZIP-arhivu, možete je otvoriti u bilo kojem pregledniku ZIP-arhiva: po otvaranju unutra će se nalaziti samo jedan vršni direktorij u kojem su dalje poddirektoriji `src` i datoteke `.classpath` i `.project`.

Jednom kad napravite arhivu, zatvorite *Eclipse*, pokrenite ga ponovno i napravite novi pomoćni *workspace*. Provjerite možete li importati napravljeni projekt i to izravno iz ZIP arhive. Potrebni koraci za import u ovom su slučaju sljedeći.

`File → Import... → General → Projects from Folder or Archive`, u prozoru koji se potom otvori klik na gumb "`Archive...`", i odaberite napravljenu ZIP arhivu (na isti ćete način importati domaće zadaće koje ćete trebati recenzirati). Nakon analize arhive, prikazat će se što se može importati. Vjerojatno će se pojaviti dvije stavke: jedna koja predstavlja korijen ZIP arhive, te druga koja predstavlja direktorij s projektom (te će desno od toga i pisati `Eclipse project`) – ostavite samo to označeno i dovršite import. Postupak će sam, čim odaberete arhivu, istu ekspandirati u *workspace* direktorij (primjerice, ako je zip arhiva `hw01-0000000000.zip`, napravit će se direktorij `hw01-0000000000.zip_expanded`, i to će postati i naziv projekta u *Project Exploreru*). Ako na disku želite smislenije ime, možete sami u *workspace* direktoriju odkomprimirati arhivu pod "normalnim" imenom (npr. `hw01-0000000000`), i potom napraviti izravno import postojećeg projekta kroz prikladnog "čarobnjaka".

Sav kod mora imati prikladan javadoc – proučite u knjizi taj dio.

Rješenje predajete na Ferka (idete na stranicu kolegija, pa *Komponente*, *Domaće zadaće*, *1. domaća zadaća*, *Upload*). Trebate uploadati ZIP-arhivu; nikako RAR-arhivu, 7z-arhivu ili nešto četvrto. Nakon što ste sigurni da je to konačno rješenje, upload morate zaključati, nakon čega više neće biti moguće raditi novi upload. Predaje koje nisu zaključane, smatraju se nepredanima. Rok za upload i zaključavanje je četvrtak u 7:00:00 ujutro.