Izpit pri predmetu Programiranje 1

1 Splošna pravila

- Na izpit se pravočasno prijavite. Če zamudite rok za prijavo, boste morali počakati na naslednji rok.
- Kdor bo manjkal na izpitu, ne da bi se pravočasno odjavil, bo ocenjen negativno. Če zamudite rok za odjavo prek sistema Studis, se lahko odjavite tudi po e-pošti (`luka.fuerst@fri.uni-lj.si`) ali pa pri nadzorniku takoj na začetku izvajanja izpita (še preden nadzornik razdeli liste z nalogami). Odjavo po e-pošti bomo upoštevali samo v primeru, če jo bomo prejeli pred pričetkom izvajanja izpita.
- Izpit boste opravljali na računalnikih v računalniških učilnicah. Uporaba lastnih računalnikov ni dovoljena.
- S seboj prinesite študentsko izkaznico.
- Na izpitu lahko uporabljate poljubne papirnate vire (npr. svoje zapiske, natisnjene prosojnice, knjige ...). Seveda lahko prinesete tudi pisalo in prazne liste papirja. Ti pripomočki vam bodo bržkone celo bolj koristili.
- USB-ključki in drugi nosilci podatkov niso dovoljeni.
- Uporabljate lahko poljubno programsko opremo, ki je nameščena na fakultetnih računalnikih, a na lastno odgovornost. Morebitne težave z razvojnimi orodji boste reševali sami. Dodatnih programov ali njihovih delov (vključkov/dodatkov, angl. plugins/add-ons) ne smete nameščati.
- Pričakujemo, da se boste na računalnik znali prijaviti, da boste znali naložiti datoteke z Učilnice in na Učilnico in da se boste znašli v datotečnem sistemu računalnika. Če niste obiskovali vaj in zato teh veščin niste pridobili, se boste morali na izpitu znajti po svoje. Pomagali vam bomo le v primeru objektivnih težav (vsake toliko časa kateri od računalnikov hočeš nočeš odpove poslušnost).
- Uporaba interneta je omejena na Učilnico in uradno javansko dokumentacijo.[1] Do dokumentacije lahko dostopate tudi tako, da stran z opisom razreda poiščete s pomočjo spletnega iskalnika: vnesite iskalni niz

  Razred `java` *različica*

  (npr. `HashMap java 18`) in izberite povezavo na stran, ki pripada uradni (Oraclovi) dokumentaciji. Kakršnakoli drugačna uporaba interneta je strogo prepovedana in se bo obravnavala kot poskus goljufije.
- Med izpitom ne odgovarjamo na vprašanja v smislu »Kaj to pomeni?«, »Ali to prav razumem?« ipd. Razumevanje naloge je sestavni del izpita. Odzvali se bomo le v

[1] `https://docs.oracle.com/en/java/`

primeru, če boste utemeljeno posumili, da ste odkrili napako. (Vsak izpit večkrat preverimo, morebitnih napak pa kljub temu ne moremo povsem izključiti.)

- For non-Slovenian speakers: upon request, you may receive both the Slovenian and English version of the exam. However, you have to notify me via email at least full 24 hours prior to the beginning of the exam.

2 O izpitu

- Čas za reševanje je omejen na 90 minut.
- Izpit je sestavljen iz štirih nalog. Vse naloge so rešljive z znanjem in izkušnjami, ki bi si jih morali pridobiti z obiskovanjem predavanj in vaj, z branjem knjige Java od začetka ter s samostojnim reševanjem domačih in dodatnih nalog ter nalog v knjigi.
- Trudimo se, da so izpitni roki v okviru istega študijskega leta po težavnosti karseda enakovredni, žal pa tega nikoli ni mogoče v popolnosti zagotoviti (tudi zato, ker je težavnost subjektiven pojem). Nikakor pa vam ne moremo jamčiti, da se vam bodo vse naloge v okviru istega izpitnega roka zdele enako težke. Za nameček upoštevajte, da so naloge urejene po temah, ne po težavnosti. To pomeni, da se vam bo lahko zdela tudi četrta naloga najlažja, prva pa najtežja.
- Pri nekaterih nalogah je lahko definiranih več skupin testnih primerov. Če se vam bo zdelo (pre)težko najti rešitev, ki pravilno obravnava vse skupine testnih primerov, se vam splača najprej osredotočiti na najlažjo skupino testnih primerov in jo šele nato posplošiti na ostale skupine.
- Naloge po vrsti obravnavnajo sledeče teme:

  (1) krmilni konstrukti, metode (poglavja 1–4 v knjigi Java od začetka);

  (2) tabele (poglavje 5);

  (3) razredi in objekti, dedovanje (poglavja 6–7);

  (4) generiki, vmesniki, vsebovalniki, lambde (poglavja 8–11).

  Prva naloga je torej rešljiva brez uporabe tabel, seveda pa to ne pomeni, da jih ne smete uporabljati (razen če bi jih izrecno prepovedali). Jasno je, da vsaka naloga vključuje tudi snov vseh prejšnjih nalog (npr. pri tretji nalogi pridejo v poštev tudi zanke in tabele).
- Na izpitu boste prejeli navodila (na papirju) in datoteko `.zip` (na Učilnici), zaščiteno z geslom. Datoteko boste prenesli na računalnik. Ko bo nadzornik razdelil liste z nalogami, bo geslo obelodanil. Datoteko boste razpakirali in prejeli imenik (mapo), ki bo vseboval podimenike `naloga1`, `naloga2`, `naloga3` in `naloga4`. Vsak podimenik bo vseboval
  - izhodiščno datoteko, ki jo boste dopolnili do rešitve naloge in oddali na Učilnico (pri prvi nalogi se bo ta datoteka imenovala `Prva.java`, pri drugi `Druga.java`, pri tretji `Tretja.java`, pri četrti nalogi pa `Cetrta.java`);
  - določeno število (tipično 10) parov testnih datotek ((`test01.in`, `test01.out`), (`test02.in`, `test02.out`), ... pri testiranju z vhodnimi datotekami oziroma (`Test01.java`, `test01.out`), (`Test02.java`, `test02.out`), ... pri testiranju s testnimi razredi);

– program za testiranje (`tj.exe`), ki ga pri testiranju z vhodi in izhodi lahko poženete kot

`tj.exe` *`Naloga`*`.java . .`

pri testiranju s testnimi razredi pa enostavno kot

`tj.exe`

- Naloga je lahko sestavljena iz več podnalog. Za vsako podnalogo je naveden delež *skritih* testnih primerov, ki se nanjo nanašajo. Delež javnih testnih primerov je približno enak, ne more pa vedno biti povsem enak. Na primer, 32% pomeni 16 od 50 skritih testnih primerov, ustrezno število javnih testnih primerov pa lahko znaša bodisi 3 bodisi 4 (od 10).
- Nekateri podatki o testnih primerih so lahko navedeni tudi v izhodiščnih datotekah, ne le na papirnih navodilih.
- Poleg metod, ki jih boste morali napisati, lahko v razrede vključite še poljubno mnogo lastnih metod in notranjih razredov. Atributi bodo v nekaterih primerih že podani, v nekaterih pa jih boste določili sami.
- V vsaki izhodiščni datoteki je že prisoten stavek

  ```
  import java.util.*;
  ```

  Če boste potrebovali še kak paket iz standardne javine knjižnice (npr. `java.util.function` ali kaj tretjega), ga mirno uvozite.
- Za veliko večino nalog velja, da ocena naloge temelji *izključno* na številu *skritih* testnih primerov, ki jih vaš program pravilno obravnava v okviru predpisane časovne omejitve (1 sekunda na testni primer). Le izjemoma se lahko pojavi naloga, ki prepoveduje rabo določenih javanskih tipov ali konstruktov; lahko, denimo, predpišemo, da morate prvo nalogo rešiti samo z uporabo tipov `int` in `boolean`. Takim nalogam se izogibamo, ne želimo pa jih povsem izključiti. Pri tovrstnih nalogah je seveda natančno predpisana kazen za uporabo prepovedanih tipov ali konstruktov (npr. prepolovitev števila točk, dobljenega na podlagi skritih testnih primerov).
- Po zaključku izpita bomo na Učilnici objavili množico skritih testnih primerov (po 50 na nalogo), na podlagi katerih se bo izračunala vaša ocena. Število točk na izpitu bomo izračunali preprosto tako, da bomo skupno število pravilno obravnavanih skritih testnih primerov delili z 2. Na primer, če vaš program pri prvi nalogi pravilno deluje za 37 skritih testnih primerov, pri drugi za 22, pri tretji za 30, pri četrti pa za 13 skritih testnih primerov, bo vaše število točk na izpitu znašalo $(37+22+30+13)/2 = 51$ (od 100 možnih). Oceno si boste torej lahko izračunali kar sami.
- Če vaš program pravilno deluje na p% javnih testnih primerov, ni nujno, da bo deloval tudi na p% skritih testnih primerov! Velike razlike med deležem pravilno obravnavanih javnih in skritih testnih primerov so sicer malo verjetne, ni pa jih mogoče povsem izključiti.

3 Splošni nasveti

- Naloge rešujte postopoma in programe sproti oddajajte na Učilnico. Na koncu šteje samo tisto, kar vam uspe v roku oddati na Učilnico.
- Svoje oddaje na Učilnico obvezno še enkrat preverite — najbolje tako, da jih ponovno prenesete na računalnik in poženete `tj.exe`. Ničkolikokrat se nam je že zgodilo, da

je kdo oddal napačno različico datoteke (morda različico iz drugega imenika), potem pa pričakoval, da ga bomo pospremili do njegovega računalnika in tam poiskali in preizkusili pravo različico. **Takim prošnjam žal ne moremo ustreči.** Še enkrat naj poudarimo, da **upoštevamo samo oddaje na Učilnici**.

- »Grda« rešitev, ki deluje, je (žal) neskončno več vredna od »lepe« rešitve, ki ne deluje.
- Da »skoraj« ni zajca ujel, pri tem izpitu še kako velja. Popolnoma pravilna (100%) in popolnoma napačna (0%) različica se lahko razlikujeta po enem znaku. Saj že vemo: **šteje izključno delovanje vaše rešitve na skritih testnih primerih**.

4 Hroščad in druge nevšečnosti

Še zdaleč ni nujno, da vam bodo vsi programi kar takoj delovali. Tudi če ste si nabrali dovolj znanja in izkušenj, je precej verjetno, da boste pozabili na kak robni pogoj, sprožili neskončno zanko, si belili glavo s trdovratno izjemo itd. **To je sestavni del izpita.** Vaša naloga je, da se postavite na lastne noge in odstranite nadležni mrčes.

Nekaj namigov:

- Samo brez panike![2]
- Program `tj.exe` poženite šele tedaj, ko utemeljeno domnevate, da program za določeno skupino testnih primerov že deluje. Do takrat pa vam bo ročno testiranje bržkone bolje služilo:

  `java` *`Naloga`* `<` `test`X`.in`

  pri testiranju z vhodnimi datotekami oziroma

  `javac Test`X`.java`
  `java Test`X

  pri testiranju s testnimi razredi. Ročno testiranje po potrebi kombinirajte z vmesnimi izpisi (več o tem v nadaljevanju).
- Ko testirate s programom `tj.exe`, naj bo poročilo `prikaz.htm` v brskalniku stalno odprto. Ob vsakem zagonu `tj.exe` ga samo osvežite (Ctrl-R).
- Če želite avtomatsko testiranje omejiti na posamezne testne primere ali pa na interval testnih primerov, uporabite stikalo `-p`:

  ```
  tj.exe -p 3
  tj.exe -p 3-3
  tj.exe -p 2-5
  ```

  V prvem primeru se testni razred, izhod vašega programa in referenčni izhod (če se razlikuje od dobljenega) zapišejo na zaslon namesto v poročilo.
- Posamezne metode razreda lahko ročno testirate tako, da v razred dodate metodo `main` (če še ni prisotna), nato pa v njej kličete izbrane metode in izpisujete njihove rezultate. Recimo, da morate v razredu *`Naloga`* napisati metodo `jeSodo`, ki preveri, ali je podano število sodo:

  ```
  public class Naloga {
  ```

[2] Douglas Adams, *Štoparski vodnik po galaksiji.*

```
    public static boolean jeSodo(int n) {
        return (n % 2 == 0);
    }
}
```

Kako metodo ročno preverimo? Preprosto: v razred *Naloga* dodamo metodo `main` ...

```
public class Naloga {

    public static void main(String[] args) {
        System.out.println(jeSodo(10));
        System.out.println(jeSodo(5));
        System.out.println(jeSodo(-3));
    }

    public static boolean jeSodo(int n) {
        return (n % 2 == 0);
    }
}
```

... nato pa program ročno prevedemo in poženemo:

```
javac Naloga.java
java Naloga
```

- Najboljše zatiralke hroščev so metode `System.out.print*`. Uporabne so za vse vrste nadlog:
  - Ta presneti `ArrayIndexOutOfBoundsException`! Nič lažjega, postavimo se v vrstico, kjer se je izjema sprožila (to lahko razberemo iz sporočila o izjemi), in pred njo vstavimo kak `System.out.println` ali dva. Lahko izpišemo samo indeks, ki je sprožil izjemo, lahko pa tudi kaj drugega. Na primer, sledeča koda lahko sproži izjemo:

    ```
    for (int i = 0; i < t.length; i++) {
        if (t[i] < 0) {
            t[i] = t[i + 1];
        }
    }
    ```

    V tem primeru bi lahko v zanki in/ali znotraj stavka `if` izpisovali vrednost spremenljivke `i`, pred zanko pa bi lahko izpisali tudi vsebino tabele `t`:

    ```
    System.out.println(Arrays.toString(t));
    for (int i = 0; i < t.length; i++) {
        System.out.println("for: i = " + i);
        if (t[i] < 0) {
            System.out.println("    if: i = " + i);
            t[i] = t[i + 1];
        }
    }
    ```

Pri testiranju s `tj.exe` se bodo izpisi seveda pojavili tudi v poročilu `prikaz.htm`. Ko napako odpravite, jih **obvezno odstranite**, sicer testni primeri ne bodo delovali!

– Se je vaš program ujel v neskončno zanko (ali vsaj sumite, da se je to zgodilo)? V takih primerih je najenostavneje, če v zanki izpisujemo vrednosti zančnega števca:

```
int i = 0;
while (i < 100) {    // neskončna zanka!
    System.out.println(i);    // tole dodamo
    i *= 2;
}
```

Če gre res za neskončno zanko, je program po dodatku stavka za izpis bolje pognati ročno, saj bo poplava izpisov lahko imela za posledico ogromno datoteko `prikaz.htm`.

– Morda niste prepričani, ali se določen kos kode sploh izvede? Preprosto:

```
if (a < b) {    // tale pogoj bi moral biti izpolnjen, a sumimo, da ni
    System.out.println("tralala");    // tole dodamo
    ...
}
```

Če program po zagonu izpiše `tralala`, je pogoj očitno bil izpolnjen, sicer pa `a` pač ni bil manjši od `b`. Če pogoj ni izpolnjen, pa bi moral biti, smo ga nekje polomili, zato pred stavek `if` vstavimo

```
System.out.printf("a = %d, b = %d%n", a, b);
```

in marširamo naprej. Skratka: **ključno je, da ne obupamo** in da vsako napako obravnavamo kot izziv, ne pa kot zaroto Usode, ki nam ne privošči, da bi izpit opravili.

– Enodimenzionalne tabele lahko izpisujemo s pomočjo metode `Arrays.toString`, večdimenzionalne pa s pomočjo metode `Arrays.deepToString`. Metodi sicer ničesar ne izpišeta, vrneta pa niz, ki ga lahko nato izpišemo.

5 Tipične zahrbtne napake

Tovrstnih napak je preveč, da bi jih lahko našteli, a jih vseeno nekaj nanizajmo:

-
```
for (...; ...; ...);
    stavek
```

Zaradi podpičja na koncu prve vrstice se zanka vrti v prazno. Tovrstnim napakam se izognemo tako, da pri stavkih `if`, `for` in `while` vedno pišemo zavite oklepaje, tudi če telo obsega en sam stavek:

```
for (...; ...; ...) {
    stavek
}
```

- ```
  for (int i = 0; i < m; i++) {
      for (int j = 0; j < n; i++) {
          ...
      }
  }
  ```

  Tovrstne (in tudi marsikatere drugačne) napake so pogosto posledica neprevidnega kopiranja.

- ```
  for (int i = 0; i <= t.length; i++) {
      t[i] = i;
  }
  ```

  `ArrayIndexOutOfBoundsException`!

- ```
  int[] a = {1, 2, 3};
  int[] b = a;
  a[0] = 7;
  ```

  Po izvedbi te kode je tudi vrednost elementa `b[0]` enaka 7. Tabelo kopiramo z operatorjem `new` in zanko ali pa z metodo `Arrays.copyOf`. (Pozor — ta metoda deluje samo za enodimenzionalne tabele!)

- ```
  boolean a = false;
  ...
  if (a = true) {
      ...
  }
  ```

  Pogoj `a = true` bo *vedno* izpolnjen, tudi če ima `a` vrednost `false`! Zakaj? Zato, ker izraz `a = true` priredi spremenljivki `a` vrednost `true`, rezultat izraza pa postane nova vrednost spremenljivke `a`. Ker je ta enaka `true`, je pogoj izpolnjen ne glede na prvotno vrednost spremenljivke `a`.

  Pravilno je seveda

  ```
  if (a == true) {
      ...
  }
  ```

  ali pa preprosto

  ```
  if (a) {
      ...
  }
  ```

Veliko uspeha!