Tekmovanje programov za igro *Herc*

1 Osnovni podatki

Vabimo vas k udeležbi na tekmovanju programov v igri *Herc*.[1] Za vas smo pripravili grafično in besedilno ogrodje, vaša naloga pa je napisati t.i. *stroj* — javanski razred za izbiro potez. Če želite sodelovati, oddajte izvorno kodo vašega stroja na spletno učilnico do **četrtka, 10. januarja 2019**, do 23:55. V nadaljevanju bomo podrobneje opisali igro in vašo nalogo.

2 Herc

Herc igramo s kompletom standardnih dvainpetdesetih kart za bridge. Karte so razdeljene v štiri skupine po trinajst, ki jim pravimo *barve*. Barve se imenujejo srce (♡), pik (♠), karo (♢) in križ (♣). V vsaki barvi si karte po padajoči vrednosti sledijo takole: A (as), K (kralj), Q (dama), J (fant), 10, 9, 8, 7, 6, 5, 4, 3, 2. As je vreden 14 točk, kralj 13, dama 12, fant 11, desetka 10, ..., dvojka pa 2 točki. Slika 1 prikazuje vse karte v barvi karo, slika 2 pa vse štiri ase.

V igri sodelujejo štirje igralci. Kot lahko vidimo na sliki 3, jih v smeri urinega kazalca označimo z indeksi 0 (spodaj levo), 1 (zgoraj levo), 2 (zgoraj desno) in 3 (spodaj desno). V prvi partiji prične partijo (odvrže prvo karto v prvem štihu) igralec 0, v drugi igralec 1, v tretji igralec 2 itd.

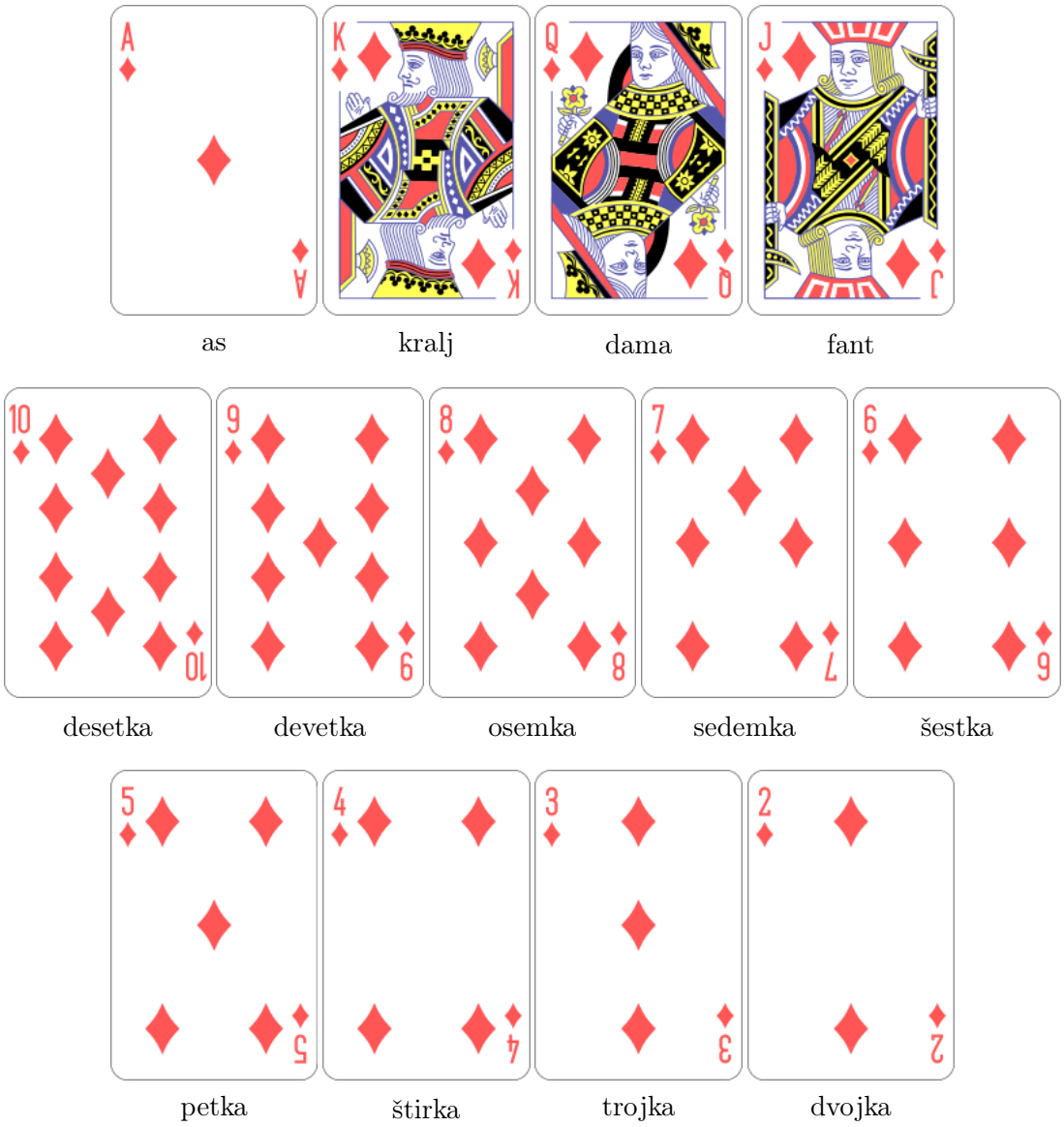
Na začetku partije vsak igralec prejme po trinajst kart. Nato igralci odigrajo trinajstih krogov, ki se formalno imenujejo *vzetki*, mi pa jim bomo skladno z uveljavljenim kvartopirskim žargonom rekli *štihi*. V vsakem štihu vsi štirje igralci na sredino mize odvržejo eno od svojih kart. Najprej odvrže karto igralec, ki *prične* štih, nato pa si igralci drug za drugim sledijo v smeri urinega kazalca. Pri tem morajo upoštevati sledeča pravila:

- Igralec, ki prične štih, lahko odvrže katerokoli od kart, ki jih ima v roki. Recimo, da je odvrgel karto barve b.
- Vsak izmed ostalih igralcev mora odvreči eno od svojih kart barve b. Če v roki nima nobene karte barve b, mora odvreči eno od kart v barvi srce. Če nima niti teh, lahko odvrže poljubno karto.

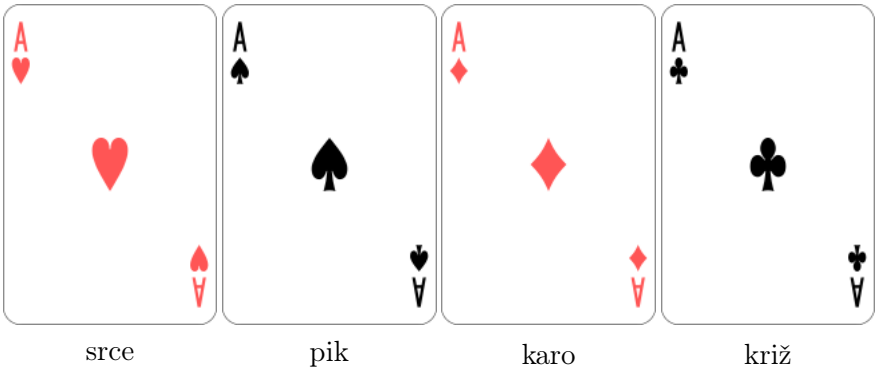
V vsakem štihu se po sledečih pravilih določi *dobitnik* štiha:

- Če je bilo v štihu odvrženo vsaj eno srce, potem štih dobi igralec, ki je odvrgel najvišjo (največ vredno) karto v barvi srce.

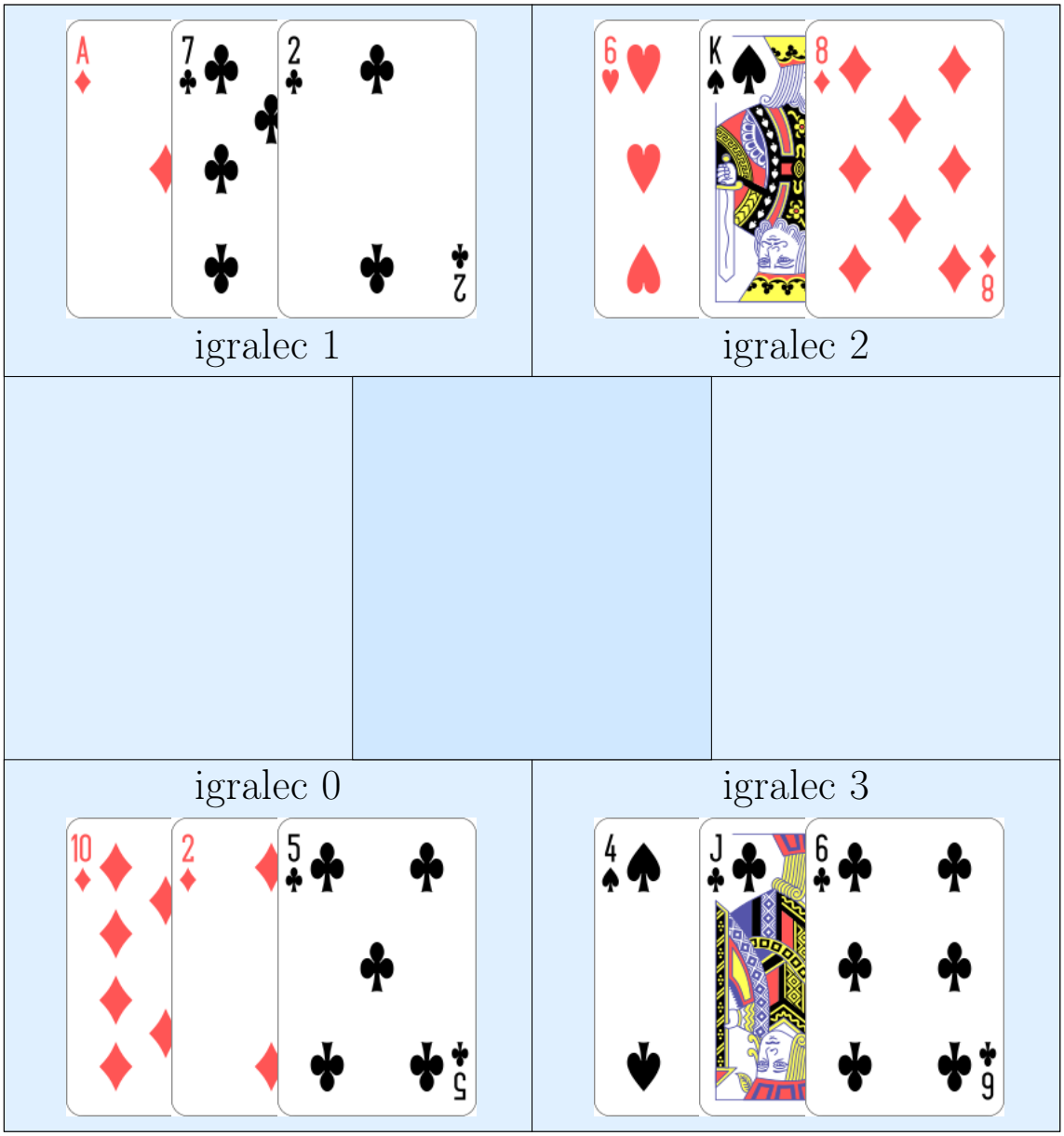
[1] Igra je sicer izmišljena, vendar pa boste kvartopirci hitro ugotovili, da gre za nekakšen presek iger s pobiranjem »štihov«, kot so tarok, šnops, bridge, preferans, whist itd. Ime je povzeto po pogovorni besedi za barvo srce (nem. Herz), ki v tej igri nastopa kot adut.



Slika 1: *Karte v barvi karo*



Slika 2: *Asi v vseh štirih barvah*



Slika 3: *Razmestitev igralcev okrog mize*

- V nasprotnem primeru dobi štih igralec, ki je odvrgel najvišjo karto v barvi b.

Dobitnik *pobere* štih (pospravi vse štiri odvržene karte v svoj zasebni kupček na mizi) in prične naslednji štih.

V primeru na sliki 3 so igralci že odigrali 10 štihov. Recimo, da enajsti štih prične igralec 0 in da odvrže ♣5. Igralec 1 lahko sedaj odvrže ♣7 ali ♣2; recimo, da se odloči za ♣2. Igralec 2 nima križev, zato je njegova edina možnost ♡6. Igralec 3 lahko odvrže ♣J ali ♣6; recimo, da odvrže ♣6. Igralec 2, ki je odvrgel najvišje (pravzaprav edino) srce, pobere štih in prične dvanajsti štih. Denimo, da odvrže ♢8. Igralec 3 nima niti kar niti src, zato lahko odvrže karkoli (recimo, da izbere ♠4). Igralec 0, denimo, izbere ♢10. Igralec 1 odvrže ♢A (edina možnost), pobere štih in prične zadnji štih. V tem štihu igralec 1 odvrže ♣7, igralec 2 ♠K, igralec 3 ♣J, igralec 0 pa ♢2. Štih pobere igralec 3.

Ko igralci odigrajo vseh trinajst štihov, se seštejejo vrednosti kart v dobljenih štihih za vse štiri igralce. Recimo, da skupna vrednost pobranih kart za igralca i (za $i \in \{0, 1, 2, 3\}$) znaša v_i. Končni rezultat partije se nato določi takole:

- Če velja $v_0 + v_2 > v_1 + v_3$, potem igralca v_0 in v_2 (oba!) prejmeta po $(v_0 + v_2 - 208)$ točk, igralca v_1 in v_3 pa ne dobita ničesar.
- V nasprotnem primeru igralca v_1 in v_3 prejmeta po $(v_1 + v_3 - 208)$ točk, igralca v_0 in v_2 pa ne dobita ničesar.

Igralca 0 in 2 se torej borita za skupno dobro proti igralcema 1 in 3 (in seveda obratno).

3 Vaša naloga

Če želite sodelovati, storite sledeče:

- S spletne učilnice prenesite pripadajoči paket zip na svoj računalnik in ga razpakirajte.
- Znotraj mape `src` izdelajte mapo `s`*XXXXXXXX*, kjer je *XXXXXXXX* vaša vpisna številka.
- V mapi[2] `src/s`*XXXXXXXX* izdelajte datoteko `Stroj_`*Ime*`.java`, kjer je *Ime* niz, ki se prične z veliko ali malo črko angleške abecede, nato pa sledi še največ 15 velikih ali malih črk angleške abecede, števk in/ali podčrtajev. Bodite izvirni!
- Povsem na začetku datoteke `Stroj_`*Ime*`.java` (torej še pred stavke `import`) zapišite vrstico

  `package s`*XXXXXXXX*`;`

  Niz *XXXXXXXX* seveda še vedno predstavlja vašo vpisno številko.
- V datoteki `Stroj_`*Ime*`.java` nato definirajte razred z imenom `Stroj_`*Ime*, ki implementira vmesnik `Stroj`. Vmesnik se nahaja v paketu `skupno` in datoteki `src/skupno/Stroj.java`.
- Rešitev lahko sestavite iz več samostojnih razredov, vendar pa morate potem vsakega od njih postaviti v mapo `src/s`*XXXXXXXX* in na začetku vsakega napisati

  `package s`*XXXXXXXX*`;`

[2] V sistemu Windows je ločilo med mapami znak \, zato se relativna pot do izdelane mape glasi `src\s`*XXXXXXXX*.

Imena morebitnih drugih razredov, ki sestavljajo vašo rešitev, se **ne smejo** pričeti s predpono `Stroj_`.

V razredu `Stroj_`*Ime* boste realizirali igralno logiko za vaš *stroj* (računalniškega igralca). Ko bo vaš stroj sodeloval v igri, bo ogrodje ob zagonu ustvarilo objekt razreda `Stroj_`*Ime*, nato pa v skladu s potekom partije nad tem objektom klicalo metode `novaPartija`, `pricetekStiha`, `sprejmiPotezo`, `izberiPotezo` in `rezultat`. Vse metode boste morali sprogramirati (četudi s praznim telesom), saj se program sicer sploh ne bo prevedel. Oglejmo si posamezne metode:

- `public void novaPartija(int polozaj, MnozicaKart karte)`

  Ogrodje pokliče to metodo ob pričetku vsake partije. Parameter `polozaj` podaja položaj (indeks) vašega stroja za mizo. Vrednost tega parametra je potemtakem 0, 1, 2 ali 3 (gl. sliko 3). Parameter `karte` podaja množico kart, ki jih vaš stroj na začetku partije »drži« v roki.

- `public void pricetekStiha(int zacetnik)`

  Ta metoda se pokliče ob pričetku vsakega štiha. Parameter `zacetnik` podaja indeks igralca, ki prične štih.

- `public void sprejmiPotezo(int akter, Karta karta)`

  Ta metoda se pokliče neposredno po tem, ko eden od strojev (toda ne vaš!) na mizo odvrže karto. Igralec z indeksom `akter` je odvrgel karto `karta`.

- `public Poteza izberiPotezo(long preostaliCas)`

  Ta metoda se pokliče, ko je vaš stroj na vrsti, da na mizo odvrže eno od svojih kart. Metoda mora v največ `preostaliCas` milisekundah vrniti potezo, ki jo stroj namerava izvesti. Če stroj prekorači časovno omejitev ali izbere neveljavno potezo, se partija takoj zaključi z zmago obeh nasprotnikov. Nasprotnika prejmeta vsak po 208 točk, torej toliko, kot če bi pobrala vse štihe.

- `public void rezultat(int[] tocke)`

  Ta metoda se pokliče ob vsakem zaključku partije. Element `tocke[`i`]` (za $i \in \{0, 1, 2, 3\}$) podaja končno število točk, ki jih za pravkar odigrano partijo prejme igralec z indeksom i. Ta metoda vam bo morda koristila pri analizi igre, sicer pa jo lahko brez škode implementirate s praznim telesom.

Razreda `Karta` in `MnozicaKart` boste našli v paketu `skupno` oziroma mapi `src/skupno`. V razredu `Karta` sta za vas najpomembnejši metodi `vrniBarvo`, ki vrne indeks barve karte `this` (0: srce; 1: pik; 2: karo; 3: križ), in `vrniVrednost`, ki vrne vrednost karte (torej celo število z intervala [2, 14]). Primere uporabe obeh razredov si lahko ogledate v datoteki `src/skupno/Test.java`. Razred lahko samostojno prevedete in poženete, in to tako, da se v terminalu premaknete v mapo `src`, nato pa odtipkate

```
javac skupno/Test.java
java skupno.Test
```

4 Razred `Stroj_Nakljucko`

Študent Fakultete za naključne študije Naključko Randomè z vpisno številko 12345678 je napisal razred `Stroj_Nakljucko` in ga lepo po pravilih postavil v paket `s12345678` in podmapo `src/s12345678`. Stroj `Nakljucko` izbira poteze naključno, vendar v skladu s

pravili. Kljub svoji preprostosti bo `Nakljucko` zanesljivo sodeloval na tekmovanju, saj nikoli ne izbere neveljavne poteze in nikoli ne prekorači časovne omejitve.

Pri izdelavi vašega stroja lahko izhajate iz razreda `Stroj_Nakljucko`. Stroj je mogoče že z majhnimi popravki precej izboljšati.

5 Ogrodje

Stroja ne morete poganjati samostojno, ampak le skupaj z ogrodjem. Ogrodje lahko prevedete tako, da se v terminalu postavite v izhodiščno mapo (tj. tisto, ki vsebuje podmapi `src` in `slike`) in izvršite sledeči ukaz:

```
javac -encoding UTF-8 -sourcepath src -d classes @javadat.txt
```

Datoteka `javadat.txt` vsebuje seznam vseh datotek `*.java` v podmapi `src`. Ko boste ustvarili svoj stroj, dodajte pot do njegove datoteke na ta seznam, sicer se stroj ne bo prevedel.

V izhodiščni mapi poženete ogrodje tako:[3]

`java -cp classes ogrodje.Herc` $igralec_0$ $igralec_1$ $igralec_2$ $igralec_3$ *opcije*

Parametri $igralec_0$, $igralec_1$, $igralec_2$ in $igralec_3$ podajajo igralce na posameznih položajih. Vsak od teh parametrov je lahko bodisi znak `-` (to pomeni, da bo na pripadajočem položaju sedel človeški igralec) bodisi ime stroja brez predpone `Stroj_`. Če enega ali več parametrov $igralec_i$ izpustite, bodo na manjkajočih položajih sedeli ljudje. Na primer, ukaz

```
java -cp classes ogrodje.Herc - Nakljucko
```

bo pričel partijo, v kateri bo na položaju 0 igral človek, na položaju 1 stroj `Nakljucko`, na položaju 2 in 3 pa spet človeka.

Parameter *opcije* je načeloma lahko poljubna kombinacija sledečega, čeprav, kot bomo videli, vse kombinacije niso smiselne:

- `-t` *sekunde*: Časovna omejitev (v sekundah) za vse stroje. Privzeta vrednost znaša 10. Vrednost 0 pomeni, da ni časovne omejitve.
- `-r` *datoteka*: Razporeditev kart se bo v vsaki partiji prebrala iz datoteke *datoteka*, namesto da bi se tvorila naključno. Vsaka vrstica v datoteki podaja eno razporeditev. V prvi partiji se bo uporabila prva razporeditev, v drugi druga itd., ko razporeditev zmanjka, pa bo program za naslednjo partijo spet uporabil prvo razporeditev iz datoteke. Vsaka razporeditev mora biti opisana z nizom

  $karte_0$ `|` $karte_1$ `|` $karte_2$ `|` $karte_3$

  pri čemer je $karte_i$ s presledki ločen seznam kart za igralca z indeksom i. Vsaka karta mora biti podana s parom znakov bv, kjer b predstavlja barvo (`s`: srce; `p`: pik; `a`: karo; `r`: križ), v pa vrednost (eden od znakov `A`, `K`, `Q`, `J`, `T` (desetka), `9`, `8`, `7`, `6`, `5`, `4`, `3` in `2`). Kot primer tovrstne datoteke služi datoteka `razporeditev.txt` v izhodiščni mapi.
- `-s` *seme*: Seme (pozitivno celo število) generatorja naključnih števil, ki se uporablja za razporejanje kart med igralce. Ta parameter se upošteva le v primeru, če parameter `-r` ni prisoten. Pri istem semenu se bo v vsakem zagonu programa tvorilo enako zaporedje razporeditev v zaporedju partij.

[3] Če ogrodje zaganjate iz podmape `classes`, je parameter `-cp classes` odveč.

- `-p` *igralec*: Indeks igralca (0, 1, 2 ali 3), ki bo pričel prvo partijo. Privzeta vrednost tega parametra je 0.
- `-h`: Če je ta parameter prisoten, bodo v grafičnem načinu razkrite samo karte v rokah človeških igralcev.
- `-b`: Program bo tekel v besedilnem načinu. Besedilni način je namenjen predvsem za samodejno igranje strojev med sabo.
- `-n` *številoPartij*: Število partij, ki naj se odigrajo v besedilnem načinu. V grafičnem načinu parameter nima učinka.
- `-q`: Če program teče v besedilnem načinu in če med seboj igrajo štirje stroji, potem ob prisotnosti tega parametra ogrodje na zaslon ne bo izpisovalo ničesar, na pisanje v dnevnik pa ta parameter ne vpliva.
- `-d` *datoteka*: Če je ta parameter prisoten, se na konec datoteke *datoteka* ob zaključku vsake partije zapiše njen potek. Datoteka se samodejno ustvari, če ne obstaja. **Pozor:** če datoteka že obstaja, se ne ustvari na novo, ampak se nova vsebina zgolj doda na konec datoteke.

Na primer,

```
java -cp classes Bucko Strucko Bucko Strucko -b -n 20 -q -d dnevnik.txt
```

požene ogrodje v besedilnem načinu brez izpisov na zaslon. Stroji `Bucko`, `Strucko`, `Bucko` in `Strucko` bodo med seboj odigrali 20 partij, potek posameznih partij pa se bo izpisoval v datoteko `dnevnik.txt`. V vsaki partiji se bo razporeditev kart določila naključno. Prvi in drugi `Bucko` (ter prvi in drugi `Strucko`) sta v resnici *dva različna objekta* razreda `Stroj_Bucko` (oziroma `Stroj_Strucko`), torej dva ločena stroja, ki imata sicer skupno programsko kodo, vendar povsem ločeno stanje.

Da si prihranite nekaj tipkanja, se vam ukaz za zagon ogrodja splača shraniti v skriptno datoteko, npr. `herc.sh` na Linuxu ali Macu (ne pozabite na ukaz `chmod +x herc.sh`) oziroma `herc.bat` na Oknih.

V datoteki `cakanje.txt` lahko po želji nastavljate parametre, ki v grafičnem načinu podajajo čakalne čase (v milisekundah) po posameznih dogodkih in lahko vplivajo na udobje pri igranju oziroma spremljanju poteka igre.

6 Oddaja stroja

Na spletno učilnico oddajte datoteko `s`*XXXXXXXX*`.zip`, kjer je *XXXXXXXX* vaša vpisna številka. Paket `zip` naj vsebuje mapo `s`*XXXXXXXX*, ta pa datoteko `Stroj_`*Ime*`.java` (*Ime* je ime vašega stroja) in po potrebi še druge datoteke s končnico `.java`, ki tvorijo vašo rešitev.

7 Potek tekmovanja

Tekmovanje bo sestavljeno iz N krogov, pri čemer bo N določen po izteku roka za oddajo. Za vsak krog bomo pripravili po eno naključno razporeditev kart. V vsakem krogu se bosta za vsak par strojev A in B pri razporeditvi kart za tisti krog odigrali po dve partiji:

(1) $A : B : A : B$ (dva različna objekta razreda `Stroj_`A bosta igrala proti dvema različnima objektoma razreda `Stroj_`B, partijo pa prične prvi objekt razreda `Stroj_`A).

(2) $B : A : B : A$ (dva različna objekta razreda `Stroj_`B bosta igrala proti dvema različnima objektoma razreda `Stroj_`A, partijo pa prične prvi objekt razreda `Stroj_`B).

Časovna omejitev bo znašala 10 sekund za vsak posamezen stroj v vsaki posamezni partiji.

Po N krogih partij med strojema A in B bo kot zmagovalec tega dvoboja proglašen stroj, ki je zbral več točk (in ne nujno tudi več zmag!). Zmagovalec bo prejel 1 točko, poraženec pa 0 točk. Če oba stroja v medsebojnih partijah dosežeta enako število točk, prejme vsak stroj po 0,5 točke. Na podlagi dobljenih točk se sestavi lestvica. Če ima več strojev enako število točk z dvobojev, jih bomo razvrstili glede na skupno število osvojenih točk.

Avtor prvouvrščenega stroja bo pri predmetu Programiranje 1 prejel dodatnih 20 točk. Drugouvrščeni bo prejel 16 dodatnih točk, tretjeuvrščeni 12, četrto- in petouvrščeni po 9 in 6, šesto- do desetouvrščeni pa po 3 dodatne točke. Dodatne točke se bodo prištele točkam, dobljenim z domačih nalog. Še veliko pomembnejše od točk pa je seveda zadovoljstvo in prestiž, ki ga prinese dober rezultat.

8 Dodatna pravila

- Poraba pomnilniškega prostora bo omejena na 100 MB na stroj.
- Stroj ne sme ustvarjati niti oziroma procesov.
- Stroj ne sme ustvarjati datotek, prav tako pa ne sme vanje pisati ali iz njih brati.
- Prepovedana je uporaba ukazov za delo z grafiko in ogrodjem Swing.
- Prepovedana je uporaba razredov iz paketa `java.lang.reflect` in javanskega introspekcijskega mehanizma (angl. *reflection*) nasploh.
- Pri atributih lahko določilo `static` uporabljate izključno v kombinaciji z določilom `final`; statične spremenljivke so prepovedane. Če ogrodje ustvari več objektov vašega stroja, mora namreč vsak stroj imeti svoje ločeno stanje, zato atributi ne smejo biti statični.
- Uporabljajte kodiranje UTF-8 (brez BOM) ali pa se vzdržite rabe šumnikov (tudi v komentarjih!).
- Če se bo stroj na kakršenkoli način poskušal dokopati do podatkov o kartah v rokah drugih igralcev, bo diskvalificiran, njegov avtor pa bo obravnavan kot goljuf in temu primerno kaznovan.
- Lahko si pomagate s tiskanimi in elektronskimi viri, vendar pa mora biti sleherna vrstica programske kode vaša in samo vaša. Kdor se bo posluževal nedovoljenih oblik pomoči, bo obravnavan kot goljuf in ustrezno kaznovan.
- Za vprašanja, pripombe itd. se obrnite na `luka.fuerst@fri.uni-lj.si` ali (še raje) na forum spletne učilnice.

Veliko užitkov pri programiranju!