|  |
| --- |
| **Teme:**   * **raba vejitev,** * **parametri ukazne vrstice/glavne funkcije,** * **uporaba generatorja pseudonaključnih števil,** * **pretvorbe med primitivnimi tipi,** * **razred java.lang.Math: raba izbranih metod razreda** |

|  |
| --- |
| Opombe:  *Vse zahtevano lahko izvede z osnovnimi orodji znotraj JDK distibucije, lahko pa uporabite poljuben IDE oz. urejevalnik besedil* |

**Naloga 1**

1. Javanski program 'Katero' izpiše »prvo število je manjše« v primeru ali »drugo število je manjše« v odvisnosti od vrednosti dveh s parametrom ukazne vrstice podanih vrednosti. Spišite program.

A black background with white text

Description automatically generated

1. Javanskemu programu 'VecjeNajprej' s parametroma podate 2 številski vrednosti. Program vedno najprej izpiše večje, nato manjše število. Zapišite ustrezen program.

A black background with white text

Description automatically generatedA black background with white text

Description automatically generated

A screenshot of a computer

Description automatically generated**Naloga 2**

Javanskemu programu s parametroma podate 3 številske vrednosti. Program vedno najprej izpiše največje, nato po velikosti srednje in nazadnje najmanjše število. Zapišite ustrezen program

**Naloga 3**

Napišite program, ki bo izvedel prekodiranje zaporedne številke dneva v mesecu v ime dneva. Pri tem predpostavite, da ima mesec natanko 32 dni, ter da je prvi dan v mesecu vedno ponedeljek. Realizacija naj vključuje dva ločena programa; prvi naj zahtevano izvede s pomočjo stavkov ***if***, drugi s pomočjo stavka ***switch***. Zaporedno številko dneva v mesecu pridobite preko parametra posredovanega glavni metodi . V primeru, da je vneseno število tako, da ne more predstavljati dneva v mesecu, naj program izpiše informacijo o nepravilnem vnosu in se konča.

A black screen with white text

Description automatically generated

**Naloga 4**

Dan je naslednji javanski program:

**A screenshot of a computer

Description automatically generated**

|  |  |
| --- | --- |
| /\* Simulacija meta kocke \*/  class SimKocka{  public static void main(String[] arg){  int izbrana = (int)(Math.random()\*6+1);  if (izbrana==6){  System.out.println(" ----- ");  System.out.println("| \* \* |");  System.out.println("| \* \* |");  System.out.println("| \* \* |");  System.out.println(" ----- ");  }  }  } | Program je trenutno zmožen ugotoviti/detektirati le met številke 6. Popravite dani program tako, da bo odziv programa ustrezna grafična predstavitev meta kocke za vsako izmed števil od 1 do 5, kot je da no z obstoječo kodo na primeru meta 6. |

**Naloga 5**

A screenshot of a computer

Description automatically generatedPri določenih igrah s kocko potrebujemo za legalen met dve kocki (npr. Jamb). Vaša naloga je : 'popravite' program, da bo namesto ene 'vrgel' dve kocki. (za mazohiste in boljšo oceno : želeli bi, da se kocki za isti met izrišeta ena poleg druge. Npr. pri metu 3,5 bi želeli izris oblike :

----- -----

| \* | | \* \* |

| \* | | \* |

| \* | | \* \* |

----- -----

*(Izpis naj bo realiziran kot je podano s primerom v nalogi 4, v običajnem tekstovnem linijskem načinu.)*

**Naloga 6** (replikacija primera)

Kvadrat, z osnovnico, vzporedno z osjo koordinatnega sistema, razpenjata v ravnini 2 točki (s koordinatama x,y). Napišite program, ki ugotovi, če je izbrana točka znotraj, izven ali na robu kvadrata. Potrebne podatke za izvedbo programa v program vnesete:

1. z branjem parametrov ukazne vrstice pri zagonu programa (3 točke, prvi dve razpenjata kvadrat
2. z generatorjem pseudonaključnih števil (Math.random/0 - glej nalogo 4

*A black background with white text

Description automatically generated*

*A black background with numbers

Description automatically generated*

**Naloga 7**

Recimo, da pravokotnik v ravnini razpenjata 2 ravninski točki s celoštevilskima koordinatama. Program zajame obe točki preko argumentov ukazne vrstice, nato izpiše ploščino kvadrata, ploščino področja, ki ga omejuje največja krožnica, ki jo je mogoče vrisati v pravokotnik, ter ploščino največjega enakostraničnega trikotnika, ki lahko obstaja znotraj danega pravokotnika.

*(Verjetno boste potrebovali kakšno konstanto in kakšno funkcijo iz java.lang.Math, recimo PI in sqrt/1)*

**A computer screen shot of a program code

Description automatically generated**

**Naloga 8**

Javanski programski jezik je strogo-tipiziran programski jezik; vsak izmed primitivov (no, podatkov) je v pomnilniku videti drugačen, kar pomeni, da je npr. vrednost 15 različno binarno predstavljena v pomnilniku kadar je ta vrste integer ali kadar je vrste short. Tipično to pomeni, da podatkov ne morete kar 'mešati' med seboj. Po drugi strani pa je zapisana vsebina (15) vedno enaka (za oziroma s stališča uporabnika); posledično so določene pretvorbe med tipi (ki so take, da ne vplivajo na zapisano vrednost) avtomatično možne. Pravilo je zelo enostavno: pretvorba med vsemi številskimi tipi je samodejno možna, kadar pretvarjate iz 'manjšega' v 'večji' tip (npr. iz short v long). V primeru, da želite pretvorbo iz večjega v manjši tip, je potrebno to pretvorbo vsiliti, ker pri taki pretvorbi lahko zgubite del informacije.

1. Razvrstite vse primitivne številske tipe po velikosti, od najmanjšega do največjega. V razvrstitev umestite tudi tip char. Za vsakega od primitivnih tipov zapišite tudi število zlogov, ki jih zaseda v pomnilniku.
2. **char:**
   * Velikost: 1 bajt
   * Število zlogov: 1
3. **short:**
   * Velikost: 2 bajta
   * Število zlogov: 2
4. **int:**
   * Velikost: Običajno 4 bajti (odvisno od arhitekture računalnika)
   * Število zlogov: 4
5. **long:**
   * Velikost: Običajno 4 ali 8 bajtov (odvisno od arhitekture računalnika)
   * Število zlogov: 4 ali 8
6. **long long:**
   * Velikost: Običajno 8 bajtov
   * Število zlogov: 8
7. **float:**
   * Velikost: 4 bajti
   * Število zlogov: 4
8. **double:**
   * Velikost: Običajno 8 bajtov
   * Število zlogov: 8
9. **long double:**
   * Velikost: Lahko več kot 8 bajtov
   * Število zlogov: Odvisno od implementacije
10. V delčku programa 'Demo' je dana pretvorba med najmanjšima tipoma byte in char. Dopolnite program, da bo izvedel še vse preostale celoštevilske pretvorbe in pretvorbi v oba neceloštevilska tipa.

public class Demo{

public static void main(String[] args){

byte b=0b0001111;

System.out.println("15? :> "+b);

**short s = b**; System.out.println(s);

int i = s; System.out.println(i);  
  
A screenshot of a computer program

Description automatically generated

1. Skušajte pretvoriti double v int : double dd = 123.62; int di = dd; Ugotovite ali je pretvorba avtomatično izvedljiva. Spišite morebiten odziv prevajalnika. Preverite, če je možna vsiljena pretvorba : int di = (int)dd; in si preverite, kakšno vrednost dobite v podatku tipa int. Ali je dobljena vrednost še enaka originalni zapisani v dd?

A black background with white text

Description automatically generated

1. Tipa (vrsti) int in char sta ekvivalentni (gre za isti tip, ki pa se navzven različno predstavi). Dokažite ali ovrzite to trditev. Pomagajte si z izvedbo naslednjega zaporedja stavkov:

char c = 'a';

int ic = c; // int je char ?

System.out.println("char: "+c+" int: "+ic); // 97 je koda znaka 'a'

System.out.println("char: "+(int)c+" int: "+(char)ic); // vsilimo različen prikaz

A black background with white text

Description automatically generated

**Naloga 9**

V delu c) naloge 8 lahko opazite, da so pretvorbe iz 'večjih' tipov v 'manjše' sicer možne, vendar ne nujno brez izgube dela informacije. Kako to 'izgubo' obvladujete (jo upoštevate, vključite v pretvorbo) lahko kontrolirate z načinom pretvorbe. V poskusu bomo uporabili necela števila 5.122 , 5.5, 5.899.

1. Z vsakim izmed danih števil uporabite metodo java.lang.Math.rouns/1:

System.out.println( Math.round(5.122) );

System.out.println( Math.round(5.5) );

System.out.println( Math.round(5.899) );

Ugotovite in zapišite kako funkcija zaokroževanja deluje, ter kakšne vrste/tipa je dobljeni rezultat.  
A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

1. Izvedite enako z metodamo floor ('na tla') in ceil ('na strop').

A screenshot of a computer program

Description automatically generated

1. Ugotovite, v čem se metodi floorDiv in ceilDiv razlikujeta od metod floor in ceil. Dokazni postopek, na katerem bazira vaša ugotovitev, naj bo enak kot v predhodnih primerih.

A black background with white text

Description automatically generated

|  |
| --- |
| Realizacija spodnjih nalog ni obvezna. Programe realizirajte brez morebitnih ponavljanj (kot linearne oz. kvečjemu z rabo vejitev. |

**Naloga 10**

1. Program 'Sode' zajame dve celoštevilski vrednosti, izpiše tisto/tisti izmed njih, ki so sode
2. Program 'VsotaPozitivnih' izračuna in izpiše vsoto pozitivnih vrednosti izmed natanko 5 vnesenih
3. V program 'VeseliBiti' \*vnesemo natanko 5 števk; če je število bitov z vrednostjo 1 vseh vnesenih števk večje od polovice največjega možnega števila bitov z enicami, program izpiše 'bitov je več kot pol možnih', če jih je manj, izpiše 'postavljenih bitov je manj kot pol možnih' in če jih je enako, izpiše 'postavljenih bitov je natanko pol'.

*( če imate problem s terminom pol, lahko namesto 5 števk spišete enakovreden mehanizem s šestimi števkami )*

1. Nibble je sestavljen iz 4 bitov: v zlogu (byte) sta tako 2 nibbla, zgornji in spodnji. V podatku tipa int jih je ustrezno več. Spišite javanski program 'NibliVIntegerju', ki na zaslon izpiše desetiško vrednost največjega izmed nibblov v poljubnem vnesenem integerju (celem številu vrste int). Testno vnosno število 16909060 bi moralo tako izpisati vrednost 4.