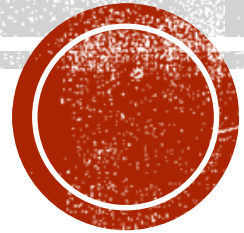
ОБЈЕКТНО ОРИЈЕНТИСАНО ПРОГРАМИРАЊЕ

ПРОГРАМСКИ ЈЕЗИК ЈАВА – 1

Коришћење класа и објеката испоручених уз JDK



ПРИСТУП СИСТЕМУ, System

- Класа System се може назвати услужном класом (енг. **utility class**).
- Неке од могућности:
  - приказ текста;
  - мерење протеклог времена;
  - покретање скупљача отпадака;
  - излазак из апликације.

ПРИКАЗ ТЕКСТА

- Класа **System** има статичко поље **System.out.**
  које представља објекат везан за стандардни излаз (попут **stdout** у језику **C**).
- System.out подразумевано показује на конзолу.
- System.err је препоручени начин за испис грешака које се јављају у току извршавања или приликом покретања програма.
- Локација исписа **System.out** и **System.err** се може променити применом метода **System.setOut(PrintStream out)** и **System.setErr(PrintStream err).**

```
System.out.print(tip arg) // исписује arg на стандардни излаз
//tip може бити: boolean, char, char[], double, float, int, long, Object, String

System.out.println(tip arg) // print + прелазак у наредни ред

System.out.printf(String format, Object...args) // форматирани испис
```

ПРИМЕР 1

- Написати Јава програм који демонстрира употребу исписа различитих типова података у форматираној и неформатираној варијанти на стандардном излазу.

ПРИМЕР 1 (2)

```
System.out.print("Пример текста");
System.out.print(" са конкатенацијом и бројем "+4);
System.out.println(); // испис празног реда
System.out.println(67.4);


double x = 26.43462;
int y = 43243;
float z = 1645.14f;
System.out.printf("x=%9.6g y=%8d z=%.3f", x, y, z);
System.out.println();

String s = "Неки текст";

char c = 'c'; // за испис знака % потребно је ставити дупли %%

System.out.printf("Стринг се умеће помоћу формата %%s овако %s", s);
System.out.printf(" док се карактер умеће помоћу формата %%%c%s", c,
System.lineSeparator());
System.out.printf("%s\t%s%n", "KPAJ", "ISPISA");
```

```
Пример текста са конкатенацијом и бројем 4
67.4
x= 26.4346 y= 43243 z=1645.140
Стринг се умеће помоћу формата %s овако Неки текст док се карактер умеће помоћу формата %c
KPAJ ISPISA
```

МЕРЕЊЕ ПРОТЕКЛОГ ВРЕМЕНА

- Мерење времена у оквиру програма може бити корисно у анализи перформанси.
- System класа омогућава два начина за мерење времена:
  - употребом метода System.currentTimeMillis();
  - употребом метода System.nanoTime().
- Оба метода враћају време протекло од Unix epoch - 1. јануар 1970 по Гриничу.
- Други метод прецизнији, јер рачуна време у наносекундама.
- Могуће индиректно мерење интервала одузимањем „апсолутних“ времена.

ПРИМЕР 2

- Написати Јава програм који мери време извршавања метода који сабира бројеве од 1 до n.
- Тестирати програм за различите вредности n,
  као и за различите методе за мерење времена
  и упоредити прецизност добијених мерења изражених у милисекундама.

ПРИМЕР 2 (2)

```
…
long pocetak1 = System.nanoTime();
long suma = sumiraj(n);
System.out.printf("Сума природних бројева до %d је %d%n", n, suma);
long kraj1 = System.nanoTime();
System.out.println("Време у ns употребом nanoTime(): " + (kraj1 - pocetak1));
System.out.printf("Време у ms употребом nanoTime(): %.5g%s",
        (kraj1 - pocetak1) / 1000000.0,System.lineSeparator());


long pocetak2 = System.currentTimeMillis();
suma = sumiraj(n);
System.out.printf("Сума природних бројева до %d је %d%n", n, suma);
long kraj2 = System.currentTimeMillis();
System.out.println("Време у ms употребом currentTimeMillis(): "
        + (kraj2 - pocetak2));
…
```

ПРИМЕР 2 (2)

```
Сума природних бројева до 10000000 је 49999995000000
Време у ns употребом nanoTime(): 10856500
Време у ms употребом nanoTime(): 10.857
Сума природних бројева до 10000000 је 49999995000000
Време у ms употребом currentTimeMillis(): 4
-----------------------------------------------
Сума природних бројева до 100000000 је 4999999950000000
Време у ns употребом nanoTime(): 22943200
Време у ms употребом nanoTime(): 22.943
Сума природних бројева до 100000000 је 4999999950000000
Време у ms употребом currentTimeMillis(): 23
-----------------------------------------------
```

ПОКРЕТАЊЕ СКУПЉАЧА ОТПАДАКА

- Јава аутоматски уклања са хипа податке који више нису у употреби, тј. на које се више не референцира, кроз тзв. систем **скупљача отпадака.**
- Скупљач отпадака се активира по потреби и његов рад је вођен хеуристикама.
- Квалитет скупљања отпадака може се оценити коришћењем два критеријума:
  1. колико је у просеку неки објекат „чекао“ у меморији од момента када је постао непотребан до момента уклањања;
  2. колики ефекат на перформансе целе виртуалне машине има рад скупљача отпадака.
- Ова два критеријума су често у колизији.
- Програмер сугерише скупљачу отпадака да се активира преко метода System.gc().
  - Ово је врло ретко оправдано радити.

ПРИМЕР 3

- Написати Јава програм који креира текст у великом броју итерација.
- Преко једне исте референцне променљиве креирају се текстуални објекти што производи велики број нереферисаних објеката током рада програма.
- Упоредити количину доступне и слободне меморије на хипу у варијантама:
  1. када скупљач сам одређује кад ће се активирати;
  2. када му корисник повремено сугерише скупљање помоћу System.gc метода.

ПРИМЕР 3 (2)

```
int n = 10000000;
int nIspis = 200000; String s;
long pocBezGC = System.nanoTime();
System.out.println( "Величине слободне меморије без позивања gc() пред испис");
for (int i = 0; i < n; i++) {
        s = new String("Текст под редним бројем " + i);
        if (i % nIspis == 0)
                stanjeMemorije();
}
System.out.printf("%nВреме без GC\t%.2f%n", (System.nanoTime() - pocBezGC) / 1e6);

long pocSaGC = System.nanoTime();
System.out.println( "Величине слободне меморије са позивањем gc() пред испис");
for (int i = 0; i < n; i++) {
        s = new String("Текст под редним бројем " + i);
        if (i % nIspis == 0) {
                System.gc();
                stanjeMemorije();
        }
}
System.out.printf("%nВреме са GC\t%.2f%n", (System.nanoTime() - pocSaGC) / 1e6);
```

ПРИМЕР 3 (3)

```
Величине слободне меморије без позивања gc() пред испис

1021MB од 1024MB | 978MB од 1024MB | 987MB од 1024MB | 973MB од 1024MB | 957MB од 1024MB | 941MB од 1024MB |
927MB од 1024MB | 1015MB од 1024MB | 999MB од 1024MB | 983MB од 1024MB | 967MB од 1024MB | 955MB од 1024MB |
939MB од 1024MB | 923MB од 1024MB | 907MB од 1024MB | 891MB од 1024MB | 879MB од 1024MB | 863MB од 1024MB |
847MB од 1024MB | 831MB од 1024MB | 815MB од 1024MB | 799MB од 1024MB | 787MB од 1024MB | 771MB од 1024MB |
755MB од 1024MB | 739MB од 1024MB | 723MB од 1024MB | 711MB од 1024MB | 695MB од 1024MB | 679MB од 1024MB |
663MB од 1024MB | 647MB од 1024MB | 631MB од 1024MB | 619MB од 1024MB | 603MB од 1024MB | 587MB од 1024MB |
571MB од 1024MB | 555MB од 1024MB | 543MB од 1024MB | 527MB од 1024MB | 511MB од 1024MB | 495MB од 1024MB |
479MB од 1024MB | 463MB од 1024MB | 451MB од 1024MB | 435MB од 1024MB | 1019MB од 1024MB | 1003MB од 1024MB |
987MB од 1024MB | 975MB од 1024MB |

Време без GC 373.09

Величине слободне меморије са позивањем gc() пред испис

55MB од 56MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB
од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од
32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB
| 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB |
31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 55MB од 56MB | 31MB
од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од
32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB | 31MB од 32MB |

Време са GC 445.92
```

ИЗЛАЗАК ИЗ АПЛИКАЦИЈЕ

- Попут функције exit у програмском језику C и Јава има метод **System.exit()**.
  - Гаси извршавање целе Јава виртуалне машине.

```
public static void exit(int status)
```

- Статус 0 указује на успешан завршетак извршавања
  док не-нула вредности указују на неуспешно извршавање, тј. на грешку.

ПРИМЕР 4

- Написати Јава програм који реализује и тестира метод за исписивање карактеристика монитора при чему су аргументи метода ширина и висина.
- Потребно је израчунати број тачака (пиксела) и сврстати монитор у категорију стандардни или широки на основу ширине и висине.
- Такође је потребно, на одговарајући начин, реаговати на некоректне уносе.

ПРИМЕР 4 (2)

```
static void karakteristikeMonitora(int sirina, int visina) {
        if (sirina <= 0 || visina <= 0) {
                System.err.println("Ширина и висина морају бити позитивни.");
                System.exit(1);
        }
        System.out.println("Ширина:\t" + sirina);
        System.out.println("Висина:\t" + visina);
        System.out.println("Број тачака:\t" + sirina * visina);
        System.out.println("Тип монитора:\t" + (sirina >= 2 * visina ? "широки"
                : "стандардни"));
}

public static void main(String[] args) {
        karakteristikeMonitora(1024, 1024);
        karakteristikeMonitora(1920, 768);
        karakteristikeMonitora(0, 230);
        // није неопходно ставити на крају програма, али не смета
        System.exit(0);
}
```

ПРИМЕР 4 (3)

```
Ширина: 1024
Висина: 1024
Број тачака: 1048576
Тип монитора: стандардни
Ширина: 1920
Висина: 768
Број тачака: 1474560
Тип монитора: широки
Ширина и висина морају бити позитивни.
```

РАД СА НИСКАМА, String

- Ниске (стрингови) представљају један од најчешће коришћених типова података.
- У програмском језику C-ниска је била представљена као низ карактера.
- Јава користи исту идеју али енкапсулира тај низ карактера у класу String.
- За разлику од C-а не захтева се да се ниска завршава терминирајућом нулом.
- Ниска, као и сваки други низ, омогућава приступ карактерима са сложеношћу O(1).

- Ниска у Јави је имутабилна (непроменљива).
  - Измена енкапсулираног низа карактера није дозвољена.
  - Било какав покушај измене објекта класе String доводи до креирања новог објекта.

КРЕИРАЊЕ КОНСТАНТНИХ НИСКИ

- Најједноставније је употребом литерала ниске.

```
String s = "Литерал ниске";
```

- Јава ће претражити тзв. списак константних ниски (енг. **string constant pool**).
  - Ако такав литерал већ постоји променљива **s** ће добити референцу ка њему.
  - Овај део меморије има специјални статус (подскуп хипа),
    јер су подаци који се у њему налазе очигледно познати већ у фази компилације.
  - Дакле, у примеру испод обе инстанцне променљиве показују на исту меморију.

```
String s1 = "Литерал ниске";
String s2 = "Литерал ниске";
```

КРЕИРАЊЕ НИСКЕ НА ХИПУ

- Са друге стране, ако се ниска креира уз експлицитну употребу оператора new, форсираће се креирање нове инстанце на хипу.
  - Без обзира што је садржај ниске познат већ у фази компилације.
  - Поред тога, креираће се литерал ниске и сместити у списак константних ниски.

```
String s1 = new String("Литерал ниске");
String s2 = new String("Литерал ниске"); // s1 и s2 показују на различито
```

- Ефективно ће бити креирана два објекта String на хипу и један ниска литерал.
- Промењљиве s1 и s2 ће референсати на објекте са хипа.
- Могуће је креирање ниске на основу низа карактера.
  - Овај случај је, попут претходног, само што не укључује креирање константне ниске.

```
char[] niz = new char[] {'к','а','р','а','к','т','е','р','и’};
String s = new String(niz);
```

- Ниска може постати позната и тек у фази извршавања (о томе касније).

ПРЕГЛЕД МЕТОДА НАД НИСКАМА

Метода	Опис
`charAt(int i)`	Враћа карактер на позицији `i`.
`compareTo(String s)`	Лексикографски упоређује ниску са прослеђеном ниском `s`.
`concat(String s)`	Надовезује ниску са прослеђеном и враћа нову ниску као повратну вредност. Сличну ствар ради и оператор `+` над нискама.
`contains(String s)`	Проверава да ли ниска садржи подниску `s`.
`endsWith(String s)`	Проверава да ли се ниска завршава са подниском `s`.
`equals(String s)`	Упоређује на једнакост ниску са прослеђеном ниском `s`.
`indexOf(char c)`	Враћа позицију првог појављивања карактера `c` или `-1` ако не постоји.
`lastIndexOf(char c)`	Враћа позицију последњег појављивања карактера `c` или `-1` ако не постоји.
`length()`	Враћа дужину ниске.

Метода	Опис
`replace(String s, String r)`	Замењује сва појављивања подниске `s` са подниском `r` и враћа тако добијену новоформирану ниску.
`split(String r)`	Раздваја ниску на низ подниски у складу са прослеђеним начином раздваја (регуларни израз `r`).
`startsWith(String s)`	Проверава да ли ниска започиње подниском `s`.
`substring(int i)`	Враћа подниску која почиње на позицији `i`. Додатно може постојати и индекс на којем се подниска завршава.
`toCharArray()`	Враћа низ карактера од којих је сачињена ниска.
`toLowerCase()`	Враћа ниску у којој су сва слова пребачена у мала.
`toUpperCase()`	Враћа ниску у којој су сва слова пребачена у велика.
`trim()`	Враћа ниску у којој је уклоњен празан простор на крајевима.

ПОРЕЂЕЊЕ НИСКИ

- Могуће је поредити ниске на два начина:
  1. испитивати да ли су једнаке,
  2. испитивати да ли је једна ниска испред друге на основу лексикографског уређења.
- За поређење да ли су две ниске једнаке користе се следећи методи.

```
public boolean equals(Object o)
public boolean equalsIgnoreCase(String s)
```

- Метод **equals** је наслеђен од класе **Object**.
- Да би две ниске биле исте:
  - морају имати исту дужину
  - и на свакој упоредној позицији морају имати исте карактере.
- Метод **equalsIgnoreCase** ради сличну ствар с тим што игнорише величину слова.

ПОРЕЂЕЊЕ НИСКИ (2)

```
String rec1 = "Ниска";
String rec2 = "Niska";
String rec3 = "нискА";
String rec4 = "Ниска";
String rec5 = new String("Ниска");

System.out.println(rec1==rec1); // true (поређење референце, увек тачно)
System.out.println(rec1==rec4); // true (константна меморија, показују на исто)
System.out.println(rec1==rec5); // false (једна у константној, друга не)
System.out.println(rec1.equals(rec2)); // false (различити карактери, писмо)
System.out.println(rec1.equals(rec3)); // false (осетљиво на величину слова)
System.out.println(rec1.equals(rec4)); // true (идентичан садржај)

System.out.println(rec1.equals(rec5)); // true (идентичан садржај)
System.out.println(rec1.equalsIgnoreCase(rec2));

//false (и даље различити карактери)
System.out.println(rec1.equalsIgnoreCase(rec3));

// true (иста слова, осим величине)
System.out.println(rec1.equalsIgnoreCase(rec4)); // true (идентичан садржај)
```

КЛАСА StringBuilder

- Непроменљивост ниски може бити проблем за ефикасно коришћење хипа.
  - У случају када је потребно вршити честе „измене“ над нискама.
  - Као што је објашњено, измене нису могуће, већ се у позадини прави нова меморија.
- Погодан начин за рад са нискама се заснива на употреби класе **StringBuilder.**
- За разлику од класе **String**, чије инстанце су непроменљиве,
  класа **StringBuilder** енкапсулира изменљиви низ карактера.
  - Њиме даље манипулише на ефикасан начин.
  - На крају се врши његова једнократна конверзија у **String** објекат.
  - **StringBuilder** нуди методе који омогућавају манипулацију унутрашњим низом карактера.
  - Корисник не мора да брине о реалокацијама (као што би то био случај у **C**-у).

ПРИМЕРИ 6 И 7

```
String s = "Тест";
int n = 10;
String sn = "";
for(int i=0; i<n; i++)
        sn+=s;
System.out.println(sn);
```

```
String s = "Тест";
int n = 10;
StringBuilder
sb=new StringBuilder();
for(int i=0; i<n; i++)
        sb.append(s);
// једнократна конверзија у ниску
String sn = sb.toString();
System.out.println(sn);
```

РАД СА ОМОТАЧИМА ПРИМИТИВНИХ ТИПОВА

- Омотачи примитивних типова су објекти који енкапсулирају примитивне типове.
- Сваки од примитивних типова
  boolean, **byte**, **short**, **char**, **int**, **long**, **float** и **double** има, редом, свој омотач тип
  Boolean, **Byte**, **Short**, **Character**, **Integer**, **Long**, **Float** и **Double**.
- Главни разлог постојања омотача је то што Јава преферира реализацију метода, класа и осталих концепата базираних на објектима.

ОБМОТАВАЊЕ

- Обмотавање (енг. boxing) се врши позивом одговарајућег конструктора.
- На пример, класа Integer има конструктор који прихвата променљиву типа int.
- Алтернативно, обмотавање је могуће и употребом статичког метода valueOf.
- Почев од Јаве 5 могуће је и аутоматско обмотавање.

```
int x=242;
Integer y = new Integer(x); // обмотавање конструктором Integer
z=Integer.valueOf(x); // обмотавање статичким методом
Integer w=x; //аутоматско обмотавање
```

ОДМОТАВАЊЕ

- Супротан поступак обмотавању је одмотавање омотач типова, те добијање одговарајућих примитивних типова.
- Ово је омогућено применом одговарајућег метода објекта омотача. На пример објекат класе **Integer** поседује метод **intValue** чији је потпис дат испод.
- Почев од Јаве 5 омогућено је аутоматско одмотавање објекта.

```
Integer x=new Integer(2311);
int y=x.intValue(); // експлицитно одмотавање
int z=x; // аутоматско омотавање
```

РАД СА СКЕНЕРИМА, Scanner

- Класа Scanner омогућава учитавање текста као и његово парсирање.
- Текст може бити унет у виду ниске, са стандардног улаза, датотеке итд.
- Скенер раздваја улазни текст у складу са постављеним сепаратором који је подразумевано празан простор.
  У њему је дефинисан већи број метода за парсирање неких стандардних типова података попут целих бројева, реалних бројева, речи, целих линија текста итд.

```
public String next()
public String nextLine()
public boolean nextBoolean()
public int nextInt()
public float nextFloat()
...
```

```
public boolean hasNext()
public boolean hasNextLine()
public boolean hasNextBoolean()
public boolean hasNextInt()
public boolean hasNextFloat()
...
```

ПРИМЕР 10

- Написати Јава програм који рачуна просек реалних бројева унетих са стандардног улаза.
- На почетку уноса корисник унапред најављује колико бројева жели да унесе, након чега следи њихов унос у сваком наредном новом реду.

ПРИМЕР 10 (2)

```
java.util.Scanner skener = new java.util.Scanner(System.in);
System.out.println("Колико бројева уносите: ");
int n = skener.nextInt();
double prosek = 0;
for (int i = 0; i < n; i++) {
        System.out.println("Број " + (i + 1) + ": ");
        double b = skener.nextDouble();
        prosek += b;
}
prosek /= n;
System.out.println(String.format("Просек унетих бројева је: %.2f", prosek));
skener.close();
```

ПРИМЕР 10 (3)

```
Колико бројева уносите:
4
Број 1:
33.3
Број 2:
11
Број 3:
1000
Број 4:
12.2
Просек унетих бројева је: 264.13
```

РАД СА МАТЕМАТИЧКИМ ФУНКЦИЈАМА, Math

- Јава подржава рад са разноврсним математичким функцијама.
- Позивање математичких функција се врши статички путем услужне класе Math.

Метода	Опис
abs(double x)	Апсолутна вредност (постоји и за друге бројевне типове).
ceil(double x)	Број заокружен на горе.
cos(double x)	Косинус.
exp(double x)	e^x
floor(double x)	Број заокружен на доле.
log(double x)	Природни логаритам броја.
max(double x, double y)	Максимум два броја (постоји и за друге бројевне типове).
min(double x, double y)	Минимум два броја (постоји и за друге бројевне типове).
pow(x, y)	x^y
random()	Псеудослучајан број из [0, 1).
round(double x)	Заокружује број.
signum(double x)	Знак броја.
sin(double x)	Синус.
sqrt(double x)	Корен броја x.
toDegrees(double rad)	Пребацује радијане у степене.
toRadians(double deg)	Пребацује степене у радијане.

РАД СА ДАТУМИМА И ВРЕМЕНИМА

- Ранија подршка за рад са датумима и временом преко класа **Date** и **Calendar.**
- Препорука је да се уместо њих (због одређених проблема у раду) користе новије класе **LocalDate, LocalTime** и **LocalDateTime.**
- Још једна значајна класа је **DateTimeFormatter** која омогућава форматирање.
- Класа **LocalDate** представља датум у **ISO-8601** формату (**yyyy-MM-dd**).

```
java.time.LocalDate danas = java.time.LocalDate.now();
```

- Алтернативно се могу користити и статички методи.

```
java.time.LocalDate datum1 = java.time.LocalDate.of(2021, 11, 25);
java.time.LocalDate datum2 = java.time.LocalDate.parse("2021-11-25");
```

ПРЕГЛЕД НЕКИХ МЕТОДА ЗА РАД СА ДАТУМИМА

Метода	Опис
`atStartOfDay()`	Враћа инстанцу `LocalDateTime` класе која одговара почетку дана.
`atTime(int s, int m)`	Враћа инстанцу `LocalDateTime` која одговара прослеђеном сату и минути. Постоје и методе са прецизнијим временом, тј. додатим секундама и наносекундама.
`format(DateTimeFormatter f)`	Форматирање датума у складу са прослеђеним форматом.
`getDayOfMonth()`	Редни број дана у току месеца.
`getDayOfWeek()`	Дан у току недеље.
`getDayOfYear()`	Редни број дана у току године.
`getMonth()`	Месец у току године.
`getYear()`	Година.
`isAfter(ChronoLocalDate d)`	Да ли је датум хронолошки после датума `d`. `ChronoLocalDate` је интерфејс који између осталих имплементира и класа `LocalDate`.
`isBefore(ChronoLocalDate d)`	Да ли је датум хронолошки пре датума `d`.
`isEqual(ChronoLocalDate d)`	Да ли су датуми једнаки.
`isLeapYear()`	Да ли је преступна година.
`lengthOfMonth()`	Број дана у месецу.
`minusDays(long d)`	Враћа нову инстанцу датума која је умањена за број дана. Подржано је одузимање и других стандардних и произвољних временских јединица.
`plusMonths(long m)`	Враћа нову инстанцу датума која је повећана за број месеци.
`now()`	Статичка метода која враћа тренутни датум.
`toEpochDay()`	Враћа број дана од дана Епохе, тј. од 1. јануара 1970. године.
`withYear(int g)`	Враћа нову инстанцу датума у којој је година замењена прослеђеном. Подржано и за друге мерне јединице.

ПРИМЕР 12

- Написати Јава програм који најпре захтева од корисника да унесе месец и годину, а потом на излазу графички приказује одабрани месец у виду табеле где су колоне дани у недељи, а редови недеље у оквиру месеца.
- При том је у ћелији табеле уписан редни број дана у месецу.

ПРИМЕР 12 (2)

```
java.util.Scanner skener = new java.util.Scanner(System.in);
System.out.println("Унесите годину ");
int godina = skener.nextInt();
System.out.println("Унесите редни број месеца");
int mesec = skener.nextInt();
System.out.println("Пон.\tУто.\tСре.\tЧет.\tПет.\tСуб.\tНед.");
java.time.LocalDate dan = java.time.LocalDate.of(godina, mesec, 1);
int danUNedeljiRbr = dan.getDayOfWeek().getValue();
for (int i = 1; i < danUNedeljiRbr; i++)
        System.out.print("\t");
do {
        System.out.print(dan.getDayOfMonth() + "\t");
        danUNedeljiRbr++;
        dan = dan.plusDays(1);
        danUNedeljiRbr = dan.getDayOfWeek().getValue();
        if (danUNedeljiRbr == 1) // идемо у наредни ред
                System.out.println();
} while (dan.getDayOfMonth() != 1);
skener.close();
```

ПРИМЕР 12 (3)

```
Унесите годину
1986
Унесите редни број месеца
11
Пон.    Уто.    Сре.    Чет.    Пет.    Суб.    Нед.
                                         1       2
3       4       5       6       7       8       9
10      11      12      13      14      15      16
17      18      19      20      21      22      23
24      25      26      27      28      29      30
```

РАД СА ДАТУМИМА И ВРЕМЕНИМА (2)

- Слично се ради и са класама **LocalTime** и **LocalDateTime.**
- Када је у питању форматирање датума и времена користи се **DateTimeFormatter.**

```
java.time.LocalDateTime datum = java.time.LocalDateTime.of(2021, 11, 30, 15, 31);
String isoDatumTekst = datum.format(DateTimeFormatter.ISO_DATE_TIME);
System.out.println(isoDatumTekst); // 2021-11-30T15:31:00
```

- Или са специфично дефинисаним начином форматирања.

```
java.time.LocalDateTime datum = java.time.LocalDateTime.of(2021, 11, 30, 15, 31);
java.time.format.DateTimeFormatter specificniFormat=
            java.time.format.DateTimeFormatter.ofPattern("dd.MM.yyyy|HH:mm:ss");
String specificniDatumTekst = datum.format(specificniFormat);
System.out.println(specificniDatumTekst); //30.11.2021|15:31:00
```

РАД СА ПСЕУДОСЛУЧАЈНИМ БРОЈЕВИМА, Random

- Раније је представљена статичка методе Math.random() која генерише псеудослучајан број из [0, 1).
- Јава пружа и неке додатне могућности кроз класу Random.
- Креирање инстанце ове класе могуће је на два начина (са или без задавања тзв. семена случајности, енг. **random seed**).

```
java.util.Random gen1 = new java.util.Random();
java.util.Random gen2 = new java.util.Random();
java.util.Random gen3 = new java.util.Random(2131);
java.util.Random gen4 = new java.util.Random(2131);
System.out.println(gen1.nextInt()); // -1610654896 (вероватно другачије сваки пут)
System.out.println(gen2.nextInt()); // 1462713902 (вероватно другачије сваки пут)
System.out.println(gen3.nextInt()); // -385602027
System.out.println(gen4.nextInt()); // -385602027
```

ПРЕГЛЕД НЕКИХ МЕТОДА КЛАСЕ Random

Метода	Опис
nextBoolean()	Униформна псеудослучајна логичка вредност.
nextBytes(byte[] bajtovi)	Уписује униформне псеудослучајне бајтове у прослеђени низ.
nextDouble()	Униформни псеудослучајни број из [0, 1) у двострукој прецизности.
nextFloat()	Униформни псеудослучајни број из [0, 1) у једнострукој прецизности.
nextGaussian()	Псеудослучајни број из нормалне расподеле са очекивањем 0 и стандардном девијацијом 1.
nextInt()	Униформни псеудослучјани цео број из целог допустивог опсега.
nextInt(int maks)	Униформни псеудослучајан цео број из опсега 0 до maks (без maks).
nextLong()	Униформни псеудослучајан дугачки цео број из допустивог опсега.
setSeed(long seme)	Ажурира “семе” генератора.

ПРИМЕР 14

- Написати Јава програм који рачуна просек вредности које враћа nextGaussian().
- На улазу корисник уписује колико бројева жели да креира,
  а на излазу се исписује просек.
- Семе генератора подесити на неку фиксирану вредност.

ПРИМЕР 14 (2)

```
java.util.Scanner skener = new java.util.Scanner(System.in);
System.out.println("Унесите број бројева ");
int n = skener.nextInt();

java.util.Random gen = new java.util.Random(12345);
double prosek=0;
for(int i=0; i<n; i++) {
        double sp = gen.nextGaussian();
        prosek+=sp;
}
prosek/=n;
System.out.println("Просек је "+prosek);

skener.close();
```

ПРИМЕР 14 (3)

```
Унесите број бројева
3
Просек је 0.4498106986322034
------------------------------
Унесите број бројева
10
Просек је 0.009936086679199475
------------------------------
Унесите број бројева
100
Просек је 0.016488380698265 26
------------------------------
Унесите број бројева
100000
Просек је 6.269351781967599E-4
------------------------------
Унесите број бројева
100000000
Просек је 1.0518669106605583E-4
```

ПИТАЊА И ЗАДАЦИ

- Шта је садржано у класи System и за шта се користи та класа?
- Упоредити System.out и System.err.
- По чему су слични, по чему се разликују?
- Како се врши мерење времена извршења неког дела програма или читавог програма написаног у програмском језику Јава? Илустровати примером, који се разликује од примера датог у тексту.
- Да ли и како програмер у Јави може уклањати податке из меморије самостално? Упоредити то са начином уклањања података из меморије у програмском језику C.
- Да ли у програмском језику Јава постоји механизам прекида програма попут функције exit, која постоји у програмском језику C? Ако постоји, објаснити како се користи тај механизам и илустровати примером.
- Колико траје животни век објекта креираног у програмском језику Јава? Упоредити оператор new из програмског језика Јава и функцију malloc из програмског језика C.

ПИТАЊА И ЗАДАЦИ (2)

- Објаснити како оператор == пореди променљиве примитивног типа, а како инстанцне променљиве. Илустровати примером.
- Чему служи оператор instanceof, како се користи и каква је веза овог оператора и принципа наслеђивања у Јави? Илустровати примером.
- Да ли је следећи део кода исправан?

```
String test = "Danas je ponedeljak";
for(int i=0;i<test.length();i+=2)
        test.charAt(i) = '#';
System.out.println(test);
```

Образложити одговор. Шта је идеја датог кода?

- Написати програм који испитује да ли је ниска која се уноси са тастатуре палиндром. За ниску се каже да је палиндром ако се исто чита с лева на десно и с десна на лево. Нпр. “АНА”, “123-321” и сл.
- Примером илустровати и објаснити разлике између метода equals и equalsIgnoreCase.

ПИТАЊА И ЗАДАЦИ (3)

- Истражити класу StringBuilder и упоредити је са класом String. Илустровати примером.
- Шта су омотачи примитивних типова и за шта се користе? Примером илустровати основне карактеристике омотача типова. Како је функционисало аутоматско обмотавање и одмотавање пре Јаве 5?
- Написати Јава програм који из текста (ниске) извлачи информације о броју учесника, просечном броју освојених поена, минималном броју освојених поена и максималном броју освојених поена, ако је познато да се текст састоји од реченица где свака има структуирану форму "[Име учесника] [Број поена]".
- Написати Јава програм који рачуна колико пута се мења знак целих бројева унетих са стандардног улаза. Бројеви се уносе док се не унесе нула. Нпр. за
- унос 2, 11, -369, 104, 105, 22, -25, -36, -78, -10, 14, 0 резултат је 4 промене знака.
- Написати Јава програм који за две тачке из простора R^2, чије се координате уносе са тастатуре, одређује међусобно растојање.

ПИТАЊА И ЗАДАЦИ (4)

- Написати Јава програм који најпре захтева од корисника да унесе дан, месец и годину за два датума, а потом на излазу исписује да ли први унесени датум претходи другом унесеном датуму.
- Написати Јава програм који најпре захтева од корисника да унесе сат, минуте и секунде за два временска тренутка, а потом на излазу исписује апсолутну временску разлику, изражену у секундама, између унесених тренутака.
- Коцкица за игру “Човече не љути се” баца се 10000 пута, односно 10000 пута се генерише цели број из интервала [1, 6]. Одредити која страна коцкице је “пала” највише, а која најмање пута. Да ли је ситуација иста и у новом покретању програма? Образложити.