

Zajímavé třídy

# java.lang.System

- Ve třídě System jsou nejpoužívanější následující statické atributy:
  - `System.out` – výpis na standardní výstup.
  - `System.err` – výpis na chybový výstup.
  - `System.in` – uživatelský vstup z konzole (od Java SE 5 v kombinaci se třídou `Scanner`).
- Také je zajímavá metoda `System.getProperty()`, pomocí které můžete získat celou řadu informací o operačním systému a klientovi.
  - Do této metody se předává klíč typu `String` a metoda vrátí požadovanou informaci. Příklady klíčů:
  - `os.name`, `os.version`, `user.name`, `user.home`, ...

# java.lang.Math

- Třída Math obsahuje statické metody jako `sin()`, `cos()`, `tan()`, `log()`, `abs()` apod.
- Použití je následující:

```
int cislo = -10;
```

```
int absCislo = Math.abs(cislo);
```


absCislo = 10



- Nebo když použijete:

```
import static java.lang.Math.*;
```

Díky tomu je možné používat statické metody bez prefixu třídy tak, jako kdyby se jednalo o lokální metody.



- Pak můžete výše uvedený kód přepsat takto:

```
int cislo = -10;
```

```
int absCislo = abs(cislo);
```

Bez Math.



# Generování náhodných čísel

- Tímto způsobem získáte náhodné číslo v intervalu  $<0.0, 1.0)$ :

```
double nahodneCislo = Math.random();
```

- Takto získáte náhodné celé číslo v intervalu  $<0, 100)$ :

```
int nahodneCislo = (int) (Math.random() * 100);
```

- `Math.random()` funguje, když potřebujete vygenerovat jedno náhodné číslo. Pokud byste potřebovali vygenerovat sérii náhodných čísel, pak použijete třídu `java.util.Random`:

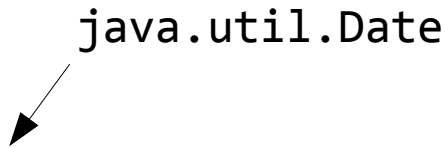
```
int nahodneCislo = new Random().nextInt(100);
```

- Poznámka: Tato třída je mnohem pokročilejší než `Math.random()`.

# Práce s datem před Java SE 8 I.

- Aktuální datum:  

```
Date currentDate = new Date();
```



- Datum zítra:  

```
Calendar calendar = new GregorianCalendar();  
calendar.add(Calendar.DAY_OF_MONTH, 1);  
Date tomorrow = calendar.getTime();
```

- Aktuální rok:  

```
Calendar calendar = new GregorianCalendar();  
int currentYear = calendar.get(Calendar.YEAR);
```

# Práce s datem před Java SE 8 II.

- Datum v textové podobě ve formátu den.měsíc.rok:

```
String stringDate = new SimpleDateFormat("dd.MM.yyyy").format(currentDate);
```

- Převod z data v textové podobě do objektu typu java.util.Date:

```
Date date = new SimpleDateFormat("dd.MM.yyyy").parse("01.06.2014");
```

- Poznámka:

- Protože tato práce s datem je velice šeredná, vznikla knihovna Joda Time, která se do Java SE 8 velice často používala. V Java SE 8 byla výrazně přepracována práce s daty a byla výrazně inspirována touto knihovnou.
- <http://www.joda.org/joda-time/>

# Práce s datem od Java SE 8 I.

- Aktuální datum:

```
LocalDate today = LocalDate.now();
```

- Datum zítra:

```
LocalDate tomorrow = today.plusDays(1);
```

- Aktuální rok:

```
int currentYear = LocalDate.now().getYear();
```

- Datum v textové podobě ve formátu den.měsíc.rok:

```
String stringDate = LocalDate.now().format(DateTimeFormatter.ofPattern("dd.MM.yyyy"));
```

- Převod z data v textové podobě do objektu typu java.time.LocalDate:

```
LocalDate date = LocalDate.parse("01.06.2014", DateTimeFormatter.ofPattern("dd.MM.yyyy"));
```

# Práce s datem od Java SE 8 II.

- Také existují třídy `LocalTime` a `LocalDateTime`.
- Také je možné provádět konverzi mezi `java.util.Date` a novými třídami.
- Další příklady:
  - <http://www.oracle.com/technetwork/articles/java/jf14-date-time-2125367.html>
  - <http://docs.oracle.com/javase/tutorial/datetime/overview/index.html>
  - <http://blog.progs.be/542/date-to-java-time>



# Regulární výrazy

- Java má out-of-the-box podporu pro regulární výrazy:
  - <http://docs.oracle.com/javase/tutorial/essential/regex/index.html>