Programowanie w języku Java (część 1)

Przemysław Klęsk

Katedra Metod Sztucznej Inteligencji i Matematyki Stosowanej
Wydział Informatyki, ZUT w Szczecinie
ul. Żołnierska 49, 71-210 Szczecin, Polska

Spis treści

Spis treści

Java

- język wysokiego poziomu zorientowany obiektowo
- programy przenaszalne (“write once, run anywhere”), niezależne od systemu operacyjnego i procesora)
- źródła `*.java` → kompilowane do kodu pośredniego `*.class`
  → wykonywane na wirtualnej maszynie Javy (JVM)
- w zamierzeniu składnia prostsza niż C, C++
- brak konieczności zwalniania pamięci przez programistę
- silna obsługa wyjątków
- liczne biblioteki gotowych klas (rozpowszechniane jako pliki `*.jar`)
- wsparcie dla wielu technologii
- wady: rozmiar kodu („poematy”), programy wolniejsze niż w językach kompilowanych natywnie, . . .

Przydatne linki

- Java (pobranie): `https://www.oracle.com/java/`
- Dokumentacja (javadocs dla wersji 8): `https://docs.oracle.com/javase/8/docs/api/`
- Eclipse (środowisko IDE): `http://www.eclipse.org/`
- Tutoriale: `https://docs.oracle.com/javase/tutorial/`, `https://www.tutorialspoint.com/java/`, ...

Spis treści

Przykład „Hello World”

- Kod źródłowy:

```
1  package kowalski.general.test;
2
3  public class MyHelloWorldClass {
4
5    public static void main(String[] args) {
6      System.out.println("Hello World.");
7    }
8
9  }
```

- Uruchomienie i wynik:

```
>> java kowalski.general.test.MyHelloWorldClass
Hello World.
```

- Elementy warte uwagi: deklaracja pakietu (`package`), ciało klasy (w tym implementacje wszystkich metod) wewnątrz klamer klasy, sygnatura metody `main(...)`.
- Konwencje nazewnicze: pakiety — małymi literami, klasy — pierwsza litera wielka + „camel case”, zmienne — pierwsza litera mała + „camel case”.

Przykład „Person”

```
1  package kowalski.general.test;
2
3  public class Person {
4
5    private String firstname, lastname;
6    private Person spouse;
7
8    public Person(String firstname, String lastname, Person spouse) {
9      this.firstname = firstname;
10     this.lastname = lastname;
11     if (spouse != null)
12       setSpouse(spouse);
13   }
14
15   public Person(String firstname, String lastname) {
16     this(firstname, lastname, null);
17   }
18
19   public void setSpouse(Person person) {
20     this.spouse = person;
21     person.spouse = this;
22   }
23
24   public String summarize() {
25     return lastname + ", " + firstname + ((spouse != null) ? " (spouse: " + spouse.lastname + ", " + spouse.
           firstname + ")" : "");
26   }
27
28   public static void main(String[] args) {
29     Person p1 = new Person("John", "Smith");
30     System.out.println("p1: " + p1.summarize());
31     Person p2 = new Person("Alice", "Smith", p1);
32     System.out.println("p1: " + p1.summarize());
33     System.out.println("p2: " + p2.summarize());
34   }
35
36 }
```

```
p1: Smith, John
p1: Smith, John (spouse: Smith, Alice)
p2: Smith, Alice (spouse: Smith, John)
```

Przykład „argumenty z linii komend”

```
1  package kowalski.general.test;
2
3  public class Person {
4
5    ...
6
7    public static void main(String[] args) {
8      Person p = new Person(args[0], args[1]);
9      System.out.println("p: " + p.summarize());
10   }
11
12 }
```

```
>> java kowalski.general.test.Person Linda Johnson
p: Johnson, Linda
```

```
>> java kowalski.general.test.Person Linda
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1
        at Person.main(Person.java:29)
```

Przykład „dziedziczenie”

```
1  package kowalski.general.test;
2
3  public class Customer extends Person {
4
5    private int id;
6
7    public Customer(String firstname, String lastname, Person spouse, int id) {
8      super(firstname, lastname, spouse);
9      this.id = id;
10   }
11
12   public String summarize() {
13     return super.summarize() + " {id: " + id + "}";
14   }
15
16   public static void main(String[] args) {
17     Customer c = new Customer("Robert", "Wallece", new Person("Emily", "Wallece"), 123);
18     System.out.println("c: " + c.summarize());
19     Person p = c;
20     p.setSpouse(new Person("Vivian", "Blue"));
21     System.out.println("p: " + p.summarize());
22   }
23
24 }
```

```
c: Wallece, Robert (spouse: Wallece, Emily) {id: 123}
p: Wallece, Robert (spouse: Blue, Vivian) {id: 123}
```

Wspólny przodek: `java.lang.Object`

`java.lang.Object`

Ważne metody:

- `String toString()` — zwraca napisową reprezentację obiektu,
- `int hashCode()` — zwraca całkowitoliczbową wartość funkcji mieszającej dla obiektu (skrót),
- `boolean equals(Object o)` — wskazuje czy inny obiekt jest „równy” danemu obiektowi,
- `void finalize()` — metoda wywoływana przez garbage collector w chwili zwracania pamięci zaalokowanej dla obiektu,
- `Object clone()` — zwraca płytką kopię danego obiektu,
- `Class<?> getClass() - zwraca klasę danego obiektu (sprawdzoną w trakcie pracy programu),`
- `...`

Przykład „metoda `toString()`”

```
1  package kowalski.general.test;
2
3  public class Person {
4
5    private String firstname, lastname;
6
7    public Person(String firstname, String lastname)  {
8      this.firstname = firstname;
9      this.lastname = lastname;
10   }
11
12   public static void main(String[] args) {
13     Person p = new Person("Jane", "Doe");
14     System.out.println(p.toString());
15     System.out.println(p);
16   }
17
18 }
```

```
Person@15db9742
Person@15db9742
```

Przykład „nadpisywanie toString()”

```
1  package kowalski.general.test;
2
3  public class Person {
4
5    private String firstname, lastname;
6
7    public Person(String firstname, String lastname)  {
8      this.firstname = firstname;
9      this.lastname = lastname;
10   }
11
12   public String toString() {
13     return lastname + ", " + firstname;
14   }
15
16   public static void main(String[] args) {
17     Person p = new Person("Jane", "Doe");
18     System.out.println(p);
19   }
20
21 }
```

```
Doe, Jane
```

```
1    ...
2    public String toString() {
3      return lastname + ", " + firstname + "(" + super.toString() + ")";
4    }
5    ...
```

```
Doe, Jane (Person@15db9742)
```

Przykład „metoda equals(...)”

```
1  public class Person {
2
3    private String firstname, lastname;
4
5    public Person(String firstname, String lastname)  {
6      this.firstname = firstname;
7      this.lastname = lastname;
8    }
9
10   public boolean equals(Object obj) {
11     Person p = (Person) obj;
12     return ((firstname.equals(p.firstname)) && (lastname.equals(p.lastname)));
13   }
14
15   public static void main(String[] args) {
16     Person p1 = new Person("Jane", "Doe");
17     Person p2 = new Person("Jane", "Doe");
18     Person p3 = new Person("Jane", "Doee");
19     System.out.println(p1.equals(p2));
20     System.out.println(p1 == p2);
21     System.out.println(p1.equals(p3));
22   }
23
24 }
```

```
true
false
false
```

Przykład „klasy czysto-obiektowe”

```java
package kowalski.general.test;

public class Person {
  ...
}
```

```java
package kowalski.general.test;

public class House {
  ...
}
```

```java
package kowalski.general.test;

public class ApplicationRunner  {

  public static void main(String[] args) {
    Person p = new Person(...);
    House h = new House(...);
    p.buy(h);
    ...
  }

}
```

Spis treści

Typy proste

Całkowitoliczbowe:

- `byte` (1 bajt)
- `short` (2 bajty)
- `int` (4 bajty)
- `long` (8 bajtów)

Zmiennoprzecinkowe (IEEE 754):

- `float` (4 bajty)
- `double` (8 bajtów)

Pozostałe:

- `char` (2 bajty, ASCII)
- `boolean` (> 1 bit, zależnie od JVM, najczęściej 1 bajt)

Istnieją obiektowe odpowiedniki (wrappery): Byte, Integer, Long, Float, Double, Character, Boolean — przydatne m.in. przy konwersjach, parsowaniu, przechowywaniu w kolekcjach, itp.

Ważne stałe dla liczb całkowitych

`java.lang.Integer`

- `BYTES` — liczba bajtów do reprezentowania liczby typu `int` w formie binarnej kodu uzupełnień do dwóch
- `MAX_VALUE` — maksymalna wartość liczby typu `int`: $2^{31} - 1$
- `MIN_VALUE` — maksymalna wartość liczby typu `int`: -2^{31}
- `SIZE` — liczba bitów do reprezentowania liczby typu `int` w formie binarnej kodu uzupełnień do dwóch

Ważne stałe dla liczb zmiennoprzecinkowych

`java.lang.Double`

- `BYTES` — liczba bajtów do reprezentowania liczby typu `double`
- `MAX_EXPONENT` — maksymalna wartość wykładnika: 1023
- `MAX_VALUE` — maksymalna wartość liczby typu `double`: $(2 - 2^{-52}) \cdot 2^{1023} \approx 1.8 \cdot 10^{308}$
- `MIN_EXPONENT` — minimalna wartość wykładnika dla *znormalizowanej* liczby typu `double`: -1022
- `MIN_NORMAL` — minimalna (niezerowa) wartość *znormalizowanej* liczby typu `double`: $2^{-1022} \approx 2.2 \cdot 10^{-308}$
- `MIN_VALUE` — minimalna (niezerowa) wartość liczby typu `double`: $2^{-1074} \approx 4.9 \cdot 10^{-324}$
- `NaN` — symbol Not-a-Number w typie `double`
- `NEGATIVE_INFINITY` — ujemna nieskończoność w typie `double`
- `POSITIVE_INFINITY` — dodatnia nieskończoność w typie `double`
- `SIZE` — liczba bitów do reprezentowania liczby typu `double`

Przykład „przewinięcie zakresu”

```
1  package kowalski.general.test;
2
3  public class NumbersTester {
4
5    public static void main(String[] args) {
6      byte x = 126;
7      System.out.println("x: " + x++);
8      System.out.println("x: " + x++);
9      System.out.println("x: " + x++);
10     System.out.println("x: " + x);
11
12     short y = (short) (Math.pow(2, 15) - 1);
13     System.out.println("y: " + y++);
14     System.out.println("y: " + y);
15
16     int z = (int) (Math.pow(2, 31) - 1);
17     System.out.println("z: " + z++);
18     System.out.println("z: " + z);
19   }
20
21 }
```

```
x: 126
x: 127
x: -128
x: -127
y: 32767
y: -32768
z: 2147483647
z: -2147483648
```

Przykład „szczególne zmienneprzecinki”

```
1  package kowalski.general.test;
2
3  public class NumberTester {
4
5    public static void main(String[] args) {
6      System.out.println("A: " + 1.0 / 0.0);
7      System.out.println("B: " + -1.0 / 0.0);
8      System.out.println("C: " + 0.0 / 0.0);
9      System.out.println("D: " + Double.NEGATIVE_INFINITY + Double.POSITIVE_INFINITY);
10     System.out.println("E: " + Math.log(0.0) + "\n");
11
12     System.out.println("F: " + Double.MAX_VALUE);
13     System.out.println("G: " + (Double.MAX_VALUE - 1));
14     System.out.println("H: " + (Double.MAX_VALUE + 1) + "\n");
15
16     System.out.println("I: " + Double.MAX_VALUE);
17     System.out.println("J: " + (Double.MAX_VALUE - Math.pow(10.0, 308 - 17)));
18     System.out.println("K: " + (Double.MAX_VALUE - Math.pow(10.0, 308 - 16)));
19     System.out.println("L: " + (Double.MAX_VALUE * 2.0));
20   }
21
22 }
```

```
A: Infinity
B: -Infinity
C: NaN
D: NaN
E: -Infinity

F: 1.7976931348623157E308
G: 1.7976931348623157E308
H: 1.7976931348623157E308

I: 1.7976931348623157E308
J: 1.7976931348623157E308
K: 1.7976931348623155E308
L: Infinity
```

Tablice jednowymiarowe

```
1  package kowalski.general.test;
2
3  import java.util.Arrays;
4
5  public class NumberTester {
6
7    public static void main(String[] args) {
8      int[] a = new int[3];
9      a[0] = 10;
10     a[1] = 20;
11     a[2] = 30;
12     int[] b = new int[] {10, 20, 30};
13
14     System.out.println("a: " + a);
15     System.out.println("a: " + Arrays.toString(a));
16     System.out.println(a.equals(b));
17     System.out.println(Arrays.equals(a, b));
18   }
19
20 }
```

```
a: [I@15db9742
a: [10, 20, 30]
false
true
```

Tablice wielowymiarowe

```java
package kowalski.general.test;

import java.util.Arrays;

public class NumberTester {

  public static void main(String[] args) {
    int[][] a = new int[2][3];
    System.out.println("a: " + Arrays.toString(a));
    System.out.println("a: " + Arrays.deepToString(a));

    int[][] b = new int[][] {{1, 2, 3}, {4, 5, 6}};
    System.out.println("b: " + Arrays.deepToString(b));
  }

}
```

```
a: [[I@15db9742, [I@6d06d69c]
a: [[0, 0, 0], [0, 0, 0]]
b: [[1, 2, 3], [4, 5, 6]]
```

Spis treści

Pętle

```
1  package kowalski.general.test;
2
3  import java.util.Arrays;
4
5  public class LoopTester {
6
7    public static void main(String[] args) {
8      int[] a = new int[10];
9      for (int i = 0; i < a.length; i++)
10       a[i] = 2 * i + 1;
11     System.out.println("a: " + Arrays.toString(a));
12
13     int[] b = new int[10];
14     int i = b.length - 1;
15     while (i > 0) {
16       b[i] = 2 * i + 1;
17       i /= 2;
18     }
19     System.out.println("b: " + Arrays.toString(b));
20
21     int[] c = new int[10];
22     i = c.length - 1;
23     do {
24       c[i] = 2 * i + 1;
25       i /= 2;
26     }
27     while (i > 0);
28     System.out.println("c: " + Arrays.toString(c));
29   }
30
31 }
```

```
a: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
b: [0, 3, 5, 0, 9, 0, 0, 0, 0, 19]
c: [0, 3, 5, 0, 9, 0, 0, 0, 0, 19]
```

Instrukcje warunkowe

```
1  package kowalski.general.test;
2
3  import java.util.Arrays;
4
5  public class ConditionTester {
6
7    public static void main(String[] args) {
8      int[] a = new int[10];
9      for (int i = 0; i < a.length; i++)
10       if (i < 4)
11         a[i] = i;
12       else if (i < 6)
13         a[i] = -i;
14       else if (i < 8)
15         a[i] = 10 * i;
16       else
17         a[i] = -10 * i;
18     System.out.println("a: " + Arrays.toString(a));
19   }
20
21 }
```

```
a: [0, 1, 2, 3, -4, -5, 60, 70, -80, -90]
```

Warunki złożone, operator trójargumentowy

```
1  package kowalski.general.test;
2
3  import java.util.Arrays;
4
5  public class ConditionTester {
6
7    public static void main(String[] args) {
8      int[] a = new int[10];
9      for (int i = 0; i < a.length; i++)
10       if (((i < 5) && (i % 2 == 0)) || (i == 9))
11         a[i] = 1;
12       else
13         a[i] = 0;
14     System.out.println("a: " + Arrays.toString(a));
15
16     a = new int[10];
17     for (int i = 0; i < a.length; i++)
18       a[i] = (((i < 5) && (i % 2 == 0)) || (i == 9)) ? 1 : 0;
19
20     System.out.println("a: " + Arrays.toString(a));
21   }
22
23 }
```

```
a: [1, 0, 1, 0, 1, 0, 0, 0, 0, 1]
```

Spis treści

Zestaw ćwiczeń 1

1. Napisz klasę w stylu „Hello World”.
2. Przeedytuj, skompiluj i uruchom klasę „Hello World” z linii komend (wykonaj wszystkie czynności poza Eclipse używając poleceń `javac` oraz `java`).
3. Napisz klasę z odpowiednią metodą `main(...)` służącą do rozwiązywania równania kwadratowego $ax^2 + bx + c = 0$. Współczynniki a, b, c mają być przekazane jako argumenty z linii komend. Wypisz na ekran rozwiązania lub informację o ich braku.
4. Stwórz klasę `Person` wyposażoną w pola: imię, nazwisko i wiek (w latach) odpowiednio dobierając typy. Nadpisz metodę `toString()`, tak aby zwracała napisową reprezentację osoby wg schematu: <imię>, <nazwisko> [<wiek>]. W metodzie `main(...)` powołaj do życia kilka obiektów `Person` i wypisz je na ekran.
5. Stwórz klasę `Customer` rozszerzającą klasę `Person` o pole identyfikator. W konstruktorze wykorzystaj wywołanie konstruktora nadklasy. Nadpisz metodę `toString()`, tak aby do napisu reprezentującego osobę dokleić z przodu identyfikator klienta — schemat: <id>: <imię>, <nazwisko> [<wiek>]. W metodzie `main(...)` powołaj do życia różne obiekty (osoby i klientów) i przetestuj ich wypisy na ekran.
6. Przypuśćmy, że tworzysz prosty edytor graficzny. Zaprojektuj klasy `Point` oraz `Polygon` reprezentujące odpowiednio punkt na płaszczyźnie oraz wielokąt. Do określenia wielokąta wykorzystaj obiekty klasy punkt. Stwórz odpowiednie metody `toString()`. Dodaj w obu klasach metodę `shift(int dx, int dy)` służącą do przesuwania obiektu na płaszczyźnie. W obu klasach nadpisz odpowiednio metody `equals(...)` (zastanów się dobrze, w jaki sposób sprawdzić, czy dwa wielokąty są identyczne).

Spis treści

Kod uzupełnień do dwóch

- Dla liczby $(b_{n-1}, b_{n-2}, \ldots, b_1, b_0)_2$ jej wartość to:

$$-b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i. \tag{1}$$

- Przykłady dla $n = 4$: $(0101)_2 = 5$, $(1011)_2 = -5$, $(0110)_2 = 6$, $(1010)_2 = -6$.
- Dodawanie:

$$\begin{aligned} (0110)_2 &= 6, \\ +(1100)_2 &= -4, \\ \hline (0010)_2 &= 2. \end{aligned}$$

$$\begin{aligned} (1010)_2 &= -6, \\ +(0100)_2 &= 4, \\ \hline (1110)_2 &= 2. \end{aligned}$$

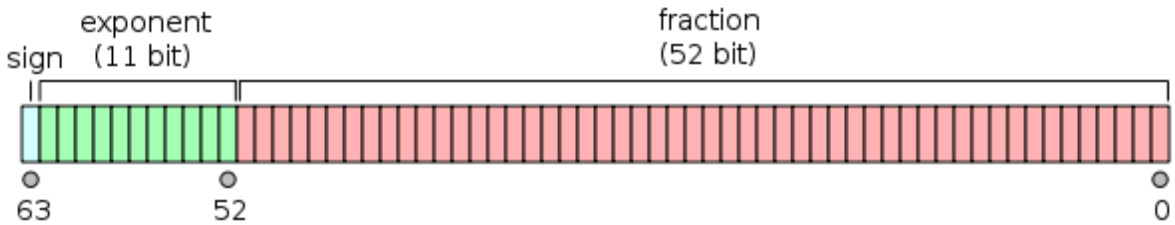
Reprezentacja zmiennoprzecinkowa

- Schemat notacji naukowej w dowolnej bazie:

$$s \cdot m \cdot B^e, \tag{2}$$

gdzie: s — znak $\in \{-1, 1\}$, m — mantysa, B — baza, a e — wykładnik.

- Przykład: $-1 \cdot 5.34721 \cdot 10^2 = -534.721$.
- Standard IEEE 754 dla podwójnej precyzji (64 bity):



$$\Big(\underbrace{b_{63}}_{s}, \underbrace{b_{62}, \ldots, b_{52}}_{e}, \underbrace{b_{51}, \ldots, b_0}_{m}\Big)_2$$

- Wartość liczby:

$$(-1)^s \cdot (1.b_{51}b_{50}, \ldots, b_0)_2 \cdot 2^{-1023+e} \tag{3}$$

$$=(-1)^s \cdot \left(1 + \sum_{i=1}^{52} b_{52-i} 2^{-i}\right) \cdot 2^{-1023+e}. \tag{4}$$

Przykład „bity bajtów”

```
1  ...
2  byte a = 25;
3  String aString = Integer.toBinaryString(a);
4  String aStringFull = String.format("%8s", Integer.toBinaryString(a & 0xff)).replace(' ', '0');
5  System.out.println("Binary representations for " + a + ": (" + aString + "), (" + aStringFull + ")");
6
7  byte b = -2;
8  String bString = Integer.toBinaryString(b);
9  String bStringFull = String.format("%8s", Integer.toBinaryString(b & 0xff)).replace(' ', '0');
10 System.out.println("Binary representations for " + b + ": (" + bString + "), (" + bStringFull + ")");
11
12 byte c = (byte) (a + b);
13 System.out.println("\nBinary operation for " + a + " + " + b + ":");
14 String cStringFull = String.format("%8s", Integer.toBinaryString(c & 0xff)).replace(' ', '0');
15 System.out.println(aStringFull);
16 System.out.println(bStringFull);
17 System.out.println("-------- [+]");
18 System.out.println(cStringFull);
```

```
Binary representations for 25: (11001), (00011001)
Binary representations for -2: (11111111111111111111111111111110), (11111110)

Binary operation for 25 + -2:
00011001
11111110
-------- [+]
00010111
```

Przykład „negacja binarna”

```
...
byte d = 10;
System.out.println("\nBinary negation example:");
String dStringFull = String.format("%8s", Integer.toBinaryString(d & 0xff)).replace(' ', '0');
System.out.println(dStringFull + " (" + d + ")");
String dStringFull2 = String.format("%8s", Integer.toBinaryString(-d & 0xff)).replace(' ', '0');
System.out.println(dStringFull2 + " (" + (-d) + ")");
String dStringFull3 = String.format("%8s", Integer.toBinaryString(~d & 0xff)).replace(' ', '0');
System.out.println(dStringFull3 + " (~" + d + ")");
String dStringFull4 = String.format("%8s", Integer.toBinaryString((~d + 0x01) & 0xff)).replace(' ', '0');
System.out.println(dStringFull4 + " (~" + d + " + 1 == " + (-d) + ")");
```

```
Binary negation example:
00001010 (10)
11110110 (-10)
11110101 (~10)
11110110 (~10 + 1 == -10)
```

Przykład „bity zmiennych przecinków”

```
1  public static String doubleToBinaryString(double x) {
2    String result = String.format("%64s", Long.toBinaryString(Double.doubleToRawLongBits(x) & 0xffffffff)).replace(' ', '0');
3    return result.substring(0, 1) + " " + result.substring(1, 12) + " " + result.substring(12);
4  }
5
6  public static void main(String[] args) {
7    double a = 1.0;
8    System.out.println("a: " + doubleToBinaryString(a) + " (" + a + ")");
9    double b = 0.5;
10   System.out.println("b: " + doubleToBinaryString(b) + " (" + b + ")");
11   double c = 1.5;
12   System.out.println("c: " + doubleToBinaryString(c) + " (" + c + ")");
13   double d = 1.375;
14   System.out.println("d: " + doubleToBinaryString(d) + " (" + d + ")");
15   double e = 1.0 + 1.0 / 3.0;
16   System.out.println("e: " + doubleToBinaryString(e) + " (" + e + ")");
17   double f = Double.POSITIVE_INFINITY;
18   System.out.println("f: " + doubleToBinaryString(f) + " (" + f + ")");
19   double g = Double.NaN;
20   System.out.println("g: " + doubleToBinaryString(g) + " (" + g + ")");
21   double h = Double.MIN_NORMAL;
22   System.out.println("h: " + doubleToBinaryString(h) + " (" + h + ")");
23   double i = Double.MIN_VALUE;
24   System.out.println("i: " + doubleToBinaryString(i) + " (" + i + ") (subnormal number)");
25   double j = -0.0;
26   System.out.println("j: " + doubleToBinaryString(j) + " (" + j + ")" + " (subnormal number)");
27 }
```

```
a: 0 01111111111 0000000000000000000000000000000000000000000000000000 (1.0)
b: 0 01111111110 0000000000000000000000000000000000000000000000000000 (0.5)
c: 0 01111111111 1000000000000000000000000000000000000000000000000000 (1.5)
d: 0 01111111111 0110000000000000000000000000000000000000000000000000 (1.375)
e: 0 01111111111 0101010101010101010101010101010101010101010101010101 (1.3333333333333333)
f: 0 11111111111 0000000000000000000000000000000000000000000000000000 (Infinity)
g: 0 11111111111 1000000000000000000000000000000000000000000000000000 (NaN)
h: 0 00000000001 0000000000000000000000000000000000000000000000000000 (2.2250738585072014E-308)
i: 0 00000000000 0000000000000000000000000000000000000000000000000001 (4.9E-324) (subnormal number)
j: 1 00000000000 0000000000000000000000000000000000000000000000000000 (-0.0) (subnormal number)
```

Spis treści

Przykład „złap wyjątek”

```
1  public static void main(String[] args) {
2    double a = Double.parseDouble("I'm not a number");
3    System.out.println("a: " + a);
4  }
```

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "I'm not a number"
        at sun.misc.FloatingDecimal.readJavaFormatString(Unknown Source)
        at sun.misc.FloatingDecimal.parseDouble(Unknown Source)
        at java.lang.Double.parseDouble(Unknown Source)
        at BinaryTester.main(BinaryTester.java:5)
```

```
1  public static void main(String[] args) {
2    String aString = "I'm not a number";
3    try {
4      double a = Double.parseDouble(aString);
5      System.out.println("a: " + a);
6    }
7    catch (NumberFormatException nfe) {
8      System.out.println("Given input string is not a number.");
9    }
10 }
```

```
Given input string is not a number.
```

Przykład „czytaj plik i parsuj”

```
1  import ...
2
3  public class FileTester {
4
5    public static void main(String[] args) {
6      String filePath = "c:/temp/test.txt";
7      List<Double> numbers = new ArrayList<Double>();
8      try {
9        FileReader fr = new FileReader(filePath);
10       BufferedReader br = new BufferedReader(fr);
11       String line;
12       while ((line = br.readLine()) != null) {
13         double number = 0.0;
14         try {
15           number = Double.parseDouble(line);
16         }
17         catch (NumberFormatException e0) {
18           number = Double.NaN;
19         }
20         numbers.add(number);
21       }
22       br.close();
23       fr.close();
24     }
25     catch (FileNotFoundException e1) {
26       System.out.println("Specified file not found. Message: " + e1.getMessage());
27     }
28     catch (IOException e2) {
29       System.out.println("Input/output exception occurred while reading file. Message: " + e2.getMessage());
30     }
31     finally {
32       // do something important on any exception (close db connection, rollback transaction, etc.)
33     }
34     System.out.println("Numbers: " + numbers);
35   }
36 }
```

```
Numbers: [-3.14, 2.71, NaN, 0.5]
```

Spis treści

Zestaw ćwiczeń 2

W każdym z zadań zastosuj obsługę wyjątków.

1. Napisz klasę, która podane napisy (jako argumenty linii komend) zapisuje do pliku tekstowego jako nowe linie. Ścieżkę do pliku określ na podstawie pierwszego argumentu i pomiń go w zapisie do pliku.
2. Napisz klasę, która podane napisy (jako argumenty linii komend) konwertuje na liczby typu `double` i wpisuje je do pliku binarnego liczba za liczbą (wykorzystaj klasy: `OutputStream` i `BufferedOutputStream`). Ścieżkę do pliku określ na podstawie pierwszego argumentu i pomiń go w zapisie do pliku. Zastosuj obsługę wyjątków. W przypadku błędu konwersji na liczbę zapisz do pliku symbol `NaN`.
3. Napisz klasę, która potrafi wczytać plik przygotowany jak w poprzednim zadaniu, a liczby wypisuje na ekran w dwóch formatach: dziesiętnym i binarnym.

Spis treści

Zadanie zaliczeniowe

Konsolowy klient serwisu pogody „World Weather Online”

- Strona serwisu: `https://www.worldweatheronline.com`.
- Zapoznać się z możliwościami responsywnego API tego serwisu (menu API), serwującego raporty pogodowe poprzez połączenia HTTP (w formatach XML i JSON).
- Wykonać darmową rejestrację otrzymując 30-dniowy klucz potrzebny do połączeń.
- Napisać konsolową aplikację w Javie będącą programem-klientem serwisu WWO, która:
  1. otrzyma parametry zapytania o pogodę z linii komend (np.: nazwa miejsca lub współrzędne geograficzne, liczba dni prognozy),
  2. nawiąże połączenie HTTP z serwisem WWO,
  3. odbierze odpowiedź w formacie XML (jedyny możliwy przy darmowej rejestracji),
  4. wyciągnie wybrane informacje pogodowe (temperatury, opady, ogólny opis) z XML-a i wyświetli dla użytkownika na konsolę.
- Wskazówki. Do połączenia HTTP przydatne będą klasy: `URL`, `HttpUrlConnection` (pakiet `java.net`). Do przesłania i odebrania bajtów poprzez połączenie przydadzą się: `DataOutputStream`, `InputStreamReader` i `BufferedReader` (pakiet `java.io`) — odczytany XML może być przechowany chwilowo jako `String`. Do parsowania XMLa i wyłuskiwania zeń informacji można użyć klas: `DocumentBuilderFactory`, `DocumentBuilder` (pakiet `javax.xml.parsers`); `InputSource` (pakiet `org.xml.sax`); oraz `Document`, `NodeList`, itp. (pakiet `org.w3c.dom`).