Politechnika Wrocławska

Java i Eclipse – krótkie wprowadzenie

dr inż. Paweł Trajdos

Politechnika Wrocławska, Katedra Systemów i Sieci Komputerowych
Wyb. Wyspianskiego 27, 50-370 Wroclaw

5 lutego 2023

Politechnika Wrocławska

Spis treści

Politechnika Wrocławska

Section 1

Java – Typy zmiennych

Politechnika Wrocławska

Typy prymitywne

Data type	Size	Default value	Range
byte	1 byte	0	-2^{8-1} - $2^{8-1}-1$
short	2 bytes	0	-2^{16-1} - $2^{16-1}-1$
int	4 bytes	0	-2^{32-1} - $2^{32-1}-1$
long	8 bytes	0	-2^{64-1} - $2^{64-1}-1$
float	4 bytes	0.0f	IEEE 754 (32 bit)
double	8 bytes	0.0d	IEEE 754 (64 bit)
boolean	1 bit*	false	false,true
char	2 bytes	'\u0000'	'\u0000' - '\uffff'

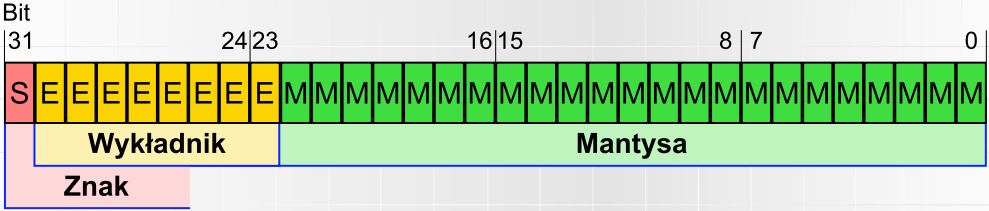
Typy prymitywne

Typy całkowitoliczbowe

Liczby całkowite są reprezentowane w kodzie uzupełnieniowym U2.
Wartość liczby: $-a_{N-1}2^{N-1} + \sum_{i=0}^{N-2} a_i 2^i$

Typy prymitywne

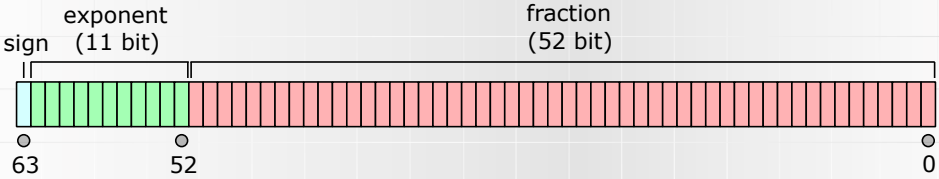
Typy zmiennoprzecinkowe IEEE 754



Wartość: $(-1)^S M * 2^E$

Typy prymitywne

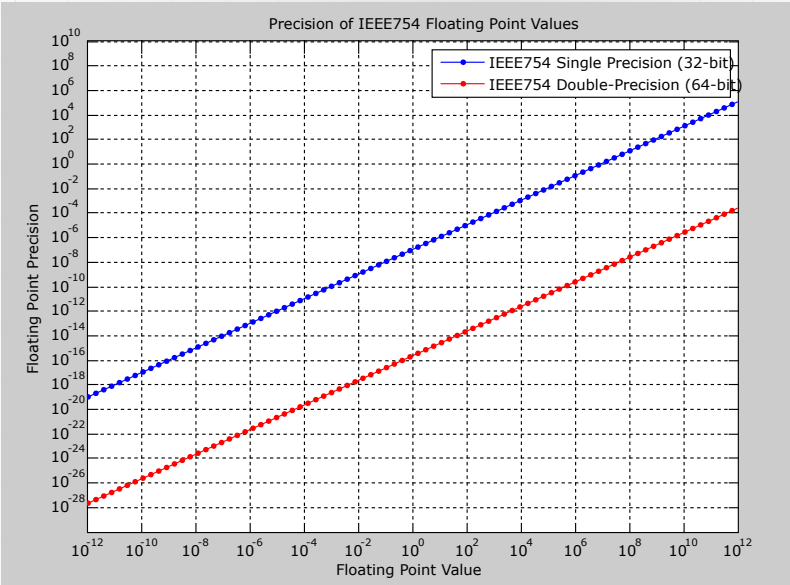
Typy zmiennoprzecinkowe IEEE 754



Wartość: $(-1)^S M * 2^E$

Typy prymitywne

Typy zmiennoprzecinkowe IEEE 754 – Precyzja !



Automatyczna konwersja typów

Convert from: \ *Convert to:*	`boolean`	`byte`	`short`	`char`	`int`	`long`	`float`	`double`
`boolean`	–	N	N	N	N	N	N	N
`byte`	N	–	Y	C	Y	Y	Y	Y
`short`	N	C	–	C	Y	Y	Y	Y
`char`	N	C	C	–	Y	Y	Y	Y
`int`	N	C	C	C	–	Y	Y*	Y
`long`	N	C	C	C	C	–	Y*	Y*
`float`	N	C	C	C	C	C	–	Y
`double`	N	C	C	C	C	C	C	–

Legenda:

- N – No conversion
- C – konwersja zwężająca (wymaga rzutowania)
- Y – automatyczna konwersja rozszerzająca
- Y* – możliwa utrata dokładności

Typy obiektowe

compact1, compact2, compact3

java.lang

Class Double

java.lang.Object
    java.lang.Number
        java.lang.Double

All Implemented Interfaces:

Serializable, Comparable<Double>

```
public final class Double
extends Number
implements Comparable<Double>
```

Typy obiektowe

	A constant holding the smallest positive normal value of type `double`, 2^{-1022}.
`static double`	**`MIN_VALUE`** A constant holding the smallest positive nonzero value of type `double`, 2^{-1074}.
`static double`	**`NaN`** A constant holding a Not-a-Number (NaN) value of type `double`.
`static double`	**`NEGATIVE_INFINITY`** A constant holding the negative infinity of type `double`.
`static double`	**`POSITIVE_INFINITY`** A constant holding the positive infinity of type `double`.
`static int`	**`SIZE`** The number of bits used to represent a `double` value.
`static Class<Double>`	**`TYPE`** The `Class` instance representing the primitive type `double`.

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.

`static long` **`doubleToRawLongBits`**`(double value)`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.

`double` **`doubleValue`**`()`

Returns the `double` value of this `Double` object.

`boolean` **`equals`**`(Object obj)`

Compares this object against the specified object.

`float` **`floatValue`**`()`

Returns the value of this `Double` as a `float` after a narrowing primitive conversion.

`int` **`hashCode`**`()`

Returns a hash code for this `Double` object.

`static int` **`hashCode`**`(double value)`

Zmienne prymitywne i obiektowe

Listing: Example16.java

```java
package examples;

import java.math.BigDecimal;

public class Example16 {
   public static void main(String[] args) {
     int a =5;
     int b =1;
     a+=b;
     a-=b;
     a*=b;
     a/=b;

     Double c = new Double(3.3);
     Double d = 4.5;
     d+=5;

     BigDecimal e  = new BigDecimal(100);
     //BigDecimal f= 1;//compiler error -- no autoboxing
     BigDecimal f = new BigDecimal(2);
     BigDecimal g = e.multiply(f);
  }
```

Biblioteka matematyczna

Listing: Example17.java

```java
package examples;

public class Example17 {
   public static void main(String[] args) {
     double a =1.3;
     double b = Math.sqrt(a);
     Double c = 2.2;
     Double d = Math.pow(a, c);
     double e = Math.random();// rnd number [0;1]
  }
}
```

Generowanie liczb losowych

Listing: Example18.java

```java
package examples;

import java.util.Random;

public class Example18 {

  public static void main(String[] args) {
     int seed = 3;
     Random rnd = new Random(seed);
     int a= rnd.nextInt(10);// {0,1,...9}
     double b = rnd.nextDouble();//[0;1]
     boolean c = rnd.nextBoolean();//coin toss
  }

}
```

Rzutowanie

Listing: Example19.java

```java
package examples;

public class Example19 {

  public static void main(String[] args) {
     int a = 1;
     //boolean b = (boolean) a;//no conversion
     double c =a;
     int d = (int) c;//casting is required
     float e = (float) c;
     double f = e;
     char g = (char) a;
     long h = (long) f;
     double i = h;
  }

}
```

Operacje na ciągach binarnych

Listing: Example20.java

```java
package examples;

public class Example20 {

  public static void main(String[] args) {
     Integer a =1;//01
     Integer b = a << 1;//10
     Integer c = 7>>1;// 011
     Integer d = c & a;//001
     int e = a |b; //11
     int f = a ^ c;//10
     int g = Integer.valueOf("1111", 2);
     System.out.println(Integer.toBinaryString(g));
  }

}
```

Instrukcje warunkowe

Listing: Example21.java

```
1  package examples;
2
3  public class Example21 {
4
5    public static void main(String[] args) {
6       boolean a=true,b=false;
7       boolean c= !a;
8       if( a && b && c) {
9         ;
10      }else if(a || b) {
11        ;
12      }else
13      {;}
14      int d = a|b? 5:-5;
15      int e = a&b? 1:2;
16   }
17
18 }
```

Instrukcje warunkowe

Listing: Example21B.java

```
1  package examples;
2
3  public class Example21B {
4
5    public static void main(String[] args) {
6       boolean a=true,b=false;
7       int x=0;
8       int z=0;
9
10      if(a || ++x==1) {;}
11      System.out.println("X: " + x);
12      if(a | ++x==1) {;}
13      System.out.println("X: " + x);
14
15      if(b && ++z==1);
16      System.out.println("Z: " + z);
17      if(b & ++z==1);
18      System.out.println("Z: " + z);
19   }
20 }
```

Switch

Listing: Example5.java

```java
package examples;

public class Example5 {

  public static void switchPrint(int n) {
    switch (n) {
    case 0:
      System.out.println("0");
      break;
    case 1:
      System.out.println("1");
      break;
    default:
      System.out.println("Something else");
    }
  }

  public static void main(String[] args) {
    Example5.switchPrint(5);
  }

}
```

Tablice i kolekcje

Listing: Example1.java

```
1  package examples;
2  import java.util.List;
3  public class Example1 {
4    public static double sum(double[] table) {
5      double sum=0;
6      for(int i=0;i<table.length;i++)sum+=table[i];
7      return sum;
8    }
9    public static Double sum2(List<Double> list) {
10     Double sum =  new Double(0);
11     for (Double double1 : list)sum+=double1;
12     return sum;
13   }
14 }
```

Tablice i kolekcje

Listing: Example1Test.java

```
1  package examples;
2  import static org.junit.Assert.*;
3  import java.util.ArrayList;
4  import java.util.List;
5  import org.junit.Test;
6  import examples.Example1;
7
8  public class Example1Test {
9    @Test
10   public void testSums() {
11     double[] array1= {1,2,3};
12     assertTrue("Sprawdzenie sumy 1", Example1.sum(array1) == 6);
13
14     List<Double> list1 = new ArrayList<Double>();
15     list1.add(new Double(1));
16     list1.add(new Double(2));
17     list1.add(new Double(3));
18
19     assertTrue("Sprawdzanie sumy 2",  Example1.sum2(list1).equals(new Double(6)));
20   }
21 }
```

Listy – operacje zaawansowane

Listing: Example10.java

```
11
12  public static void main(String[] args) {
13
14    List<String> list = new ArrayList<>(5);
15    list.add("A"); list.add("B"); list.add("C");
16    list.add(2,"D");//list.add(200,"D"); //IndexOutOfBoundException
17    System.out.println("Lista: " + list.toString() ); //A B D C
18
19    List<String> list2 = new LinkedList<String>();
20    list2.add("X"); list2.add("Y"); list2.add("Z");
21    list2.addAll(list);
22    System.out.println("Lista 2: " + list2 );// X Y Z A B D C
23
24    List<String> list3 = list2.subList(1, 3);
25    System.out.println("Lista 3: " + list3 );//  Y Z
26
27    String[] tab = {"A","C","D"};
28    List<String> list4 = Arrays.asList(tab);
29    System.out.println("Lista 4: " + list4 );
30
31    String tab2[] = (String[]) list4.toArray();
```

Listy – operacje zaawansowane

Listing: Example10.java

```
32      System.out.println("Tab2: " + Arrays.toString(tab2));
33
34      String obj = "A";
35      int idx = list2.indexOf(obj); // first occurence
36      if(idx<0)
37        System.out.println("No such element");
38      else
39        System.out.println("Element " + obj +" found at " + idx);
40
41      boolean contain = list.contains(obj);
42      if(contain) {
43        idx = list2.lastIndexOf(obj);//last occurence
44        System.out.println("Last occurence found at: " + idx);
45        list2.remove(idx);
46      }
47
```

Listy – operacje zaawansowane

Listing: Example10.java

```
49      list2.removeAll(list);
50
51      Iterator<String> iterator = list.iterator();
52      while(iterator.hasNext()) {
53        String next = iterator.next();
54        System.out.println("N: " + next);
55      }
56
57      list.sort(new Comparator<String>() {
58
59        @Override
60        public int compare(String o1, String o2) {
61          return o1.compareTo(o2);
62        }
63
64
65      });
66
67      System.out.println("Sorted: " + list);
68
```

Listy i wyrażenia lambda

Listing: Example10B.java

```
1  package examples;
2
3  import java.util.ArrayList;
4  import java.util.List;
5  import java.util.function.Consumer;
6
7  public class Example10B {
8
9    public static void main(String[] args) {
10      List<String> list = new ArrayList<>(5);
11      list.add("A"); list.add("B"); list.add("C");
12      list.add(2,"D");
13      System.out.println("Lista: " + list.toString() ); //A B D C
14      list.forEach( (n) -> {System.out.println(n);} );//lambda expression
15      Consumer<String> method = (n) -> { System.out.println(n); };
16      list.forEach(method);
17   }
18
19 }
```

Iteracja i rekurencja

Listing: Example2.java

```java
package examples;

public class Example2 {

  public static int factorial1(int n) {
    if(n<=0) return 1;
    int factorial=1;
    for(int i=1;i<=n;i++)
      factorial*=i;
    return factorial;
  }

  public static int factorial2(int n) {
    if(n<=1)return 1;
    return n*Example2.factorial2(n-1);
  }
```

Iteracja i rekurencja

Listing: Example2.java

```
18  public static int factorial3(int n) {
19    if(n<=0)return 1;
20    int factorial=1;
21    do {
22      factorial*=n--;
23    }while(n>=1);
24    return factorial;
25  }
26
27  public static int factorial4(int n) {
28    if(n<=0)return 1;
29    int factorial=1;
30    while(n>=1) {
31      factorial*=n--;
32    }
33    return factorial;
34  }
```

Iteracja i rekurencja

Listing: Example2.java

```
36  public static int factorial5(int n) {
37    if(n<=0) return 1;
38    int factorial=1;
39    for(int i=0;i<=n;i++) {
40      if(i==0)
41        continue;
42      factorial*=i;
43    }
44    return factorial;
45  }
46  public static int factorial6(int n) {
47      if(n<=0)return 1;
48      int factorial=1;
49      while(true) {
50        factorial*=n--;
51        if(n<1)
52          break;
53      }
54      return factorial;
55  }
56
57
```

Iteracja i rekurencja

Listing: Example2Test.java

```
1  package examples;
2  import static org.junit.Assert.*;
3  import org.junit.Test;
4  import examples.Example2;
5
6  public class Example2Test {
7    @Test
8    public void testFactorial1() {
9      assertTrue("Factorial 1 test", Example2.factorial1(0) == 1);
10     assertTrue("Factorial 1 test", Example2.factorial1(4) == 24);
11     assertTrue("Factorial 1 test", Example2.factorial1(3) == 6);
12     assertTrue("Factorial 1 test", Example2.factorial1(1) == 1);
13   }
14   @Test
15   public void testFactorial2() {
16     assertTrue("Factorial 2 test", Example2.factorial2(0) == 1);
17     assertTrue("Factorial 2 test", Example2.factorial2(4) == 24);
18     assertTrue("Factorial 2 test", Example2.factorial2(3) == 6);
19     assertTrue("Factorial 2 test", Example2.factorial2(1) == 1);
20   }
```

Iteracja i rekurencja

Listing: Example2Test.java

```
1   @Test
2   public void testFactorial3() {
3     assertTrue("Factorial 3 test", Example2.factorial3(0) == 1);
4     assertTrue("Factorial 3 test", Example2.factorial3(4) == 24);
5     assertTrue("Factorial 3 test", Example2.factorial3(3) == 6);
6     assertTrue("Factorial 3 test", Example2.factorial3(1) == 1);
7   }
8
9   @Test
10  public void testFactorial4() {
11    assertTrue("Factorial 4 test", Example2.factorial4(0) == 1);
12    assertTrue("Factorial 4 test", Example2.factorial4(4) == 24);
13    assertTrue("Factorial 4 test", Example2.factorial4(3) == 6);
14    assertTrue("Factorial 4 test", Example2.factorial4(1) == 1);
15  }
16 }
```

Iteracja

Labbelled break and continue

Listing: Example2B.java

```
1  package examples;
2
3  public class Example2B {
4     public static void multTable(int n) {
5       if(n<=0)
6         return;
7       rowLoop:for(int r=1;r<=n;r++) {
8         for(int c=1;c<=n;c++) {
9           System.out.println("" + r + " * " + c + "= " + r*c + " ");
10          if(c==3)break rowLoop;
11        }
12      System.out.println();
13      }
14    }
15
16    public static void main(String[] args) {
17      Example2B.multTable(5);
18    }
19
20 }
```

Iteracja

Labbelled break and continue

```
1 * 1= 1
1 * 2= 2
1 * 3= 3
```

Iteracja

Labbelled break and continue

Listing: Example2C.java

```
1  package examples;
2
3  public class Example2C {
4      public static void multTable(int n) {
5        if(n<=0)
6          return;
7        rowLoop:for(int r=1;r<=n;r++) {
8          for(int c=1;c<=n;c++) {
9            System.out.println("" + r + " * " + c + "= " + r*c + " ");
10           if(c==3)continue rowLoop;
11         }
12         System.out.println();
13       }
14     }
15
16
17   public static void main(String[] args) {
18     Example2C.multTable(5);
19   }
20 }
```

Iteracja

Labbelled break and continue

```
1 * 1= 1
1 * 2= 2
1 * 3= 3
2 * 1= 2
2 * 2= 4
2 * 3= 6
3 * 1= 3
3 * 2= 6
....
```

Tablice dwuwymiarowe – typy proste

Listing: Example4.java

```
1  package examples;
2
3  public class Example4 {
4
5    public static double[][] generateMultTable(int size){
6      double[][] table = new double[size][size];
7      for(int i=0;i<size;i++)
8        for(int j=0;j<size;j++)
9          table[i][j]= (i+1)*(j+1);
10     return table;
11   }
12
13   public static double[][] generateMultTableUpper(int size){
14     double[][] table = new double[size][];
15     for(int i=0;i<size;i++) {
16       table[i] = new double[i+1];
17       for(int j=0;j<table[i].length;j++)
18         table[i][j]=(i+1)*(j+1);
19     }
20     return table;
21   }
22
```

Tablice dwuwymiarowe – typy proste

Listing: Example4.java

```
1   public static void printMultTable(double[][] table) {
2     for(int i=0;i<table.length;i++) {
3       for(int j=0;j<table[i].length;j++)
4         System.out.print(""+table[i][j]+" ");
5       System.out.print("\n");
6     }
7   }
8
9   public static void main(String[] args) {
10    int tabSize=5;
11    double[][] tab = Example4.generateMultTable(tabSize);
12    Example4.printMultTable(tab);
13
14    tab = Example4.generateMultTableUpper(tabSize);
15    Example4.printMultTable(tab);
16  }
17
18
19 }
```

Tablice dwuwymiarowe – typy obiektowe

Listing: Example6.java

```
1  package examples;
2
3  public class Example6 {
4
5    public static Double[][] generateMultTable(int size){
6      Double[][] table = new Double[size][size];
7      for(int i=0;i<size;i++)
8        for(int j=0;j<size;j++)
9          table[i][j] = (double) ((i+1)*(j+1));
10
11     return table;
12   }
13
14   public static void printMultTable(Double[][] table) {
15
16     for(int i=0;i<table.length;i++) {
17       for(int j=0;j<table[i].length;j++)
18         System.out.print("" + table[i][j] + " ");
19       System.out.print("\n");
20     }
21
22   }
```

Tablice dwuwymiarowe – typy obiektowe

Listing: Example6.java

```
1
2   public static void main(String[] args) {
3     int size = 4;
4     Double[][] tab = generateMultTable(size);
5     printMultTable(tab);
6
7   }
8
9 }
```

Kolekcje zagnieżdżone

Listing: Example7.java

```
1  package examples;
2
3  import java.util.LinkedList;
4  import java.util.List;
5
6  public class Example7 {
7
8    public static List<List<Double>> generateMultTableUpper(int size){
9      List<List<Double>> list = new LinkedList<List<Double>>();
10     for(int i=0;i<size;i++) {
11       list.add(new LinkedList<>());
12       for(int j=0;j<=i;j++)
13         list.get(i).add((double) ((i+1)*(j+1)));
14     }
15     return list;
16   }
17
```

Kolekcje zagnieżdżone

Listing: Example7.java

```
18    public static void printMultTable(List<List<Double>> list) {
19      for(int i=0;i<list.size();i++) {
20        for(int j=0;j<list.get(i).size();j++)
21          System.out.print(""+ list.get(i).get(j)+" ");
22        System.out.print("\n");
23      }
24    }
25
26    public static void main(String[] arg) {
27      int size=4;
28      List<List<Double>> list = generateMultTableUpper(size);
29      printMultTable(list);
30
31    }
32
33  }
```

Wczytywanie danych ze strumienia

Listing: Example8.java

```
1  package examples;
2
3  import java.util.Scanner;
4
5  import java.io.*;
6
7  public class Example8 {
8
9    public static Integer readInteger(InputStream stream) {
10     Scanner scan = new Scanner(stream);
11     Integer value =new Integer(scan.nextInt());
12     scan.close();
13     return value;
14   }
```

Wczytywanie danych ze strumienia

Listing: Example8.java

```
1
2       System.out.println("Podaj liczbe:");
3       Integer value = readInteger(System.in);
4       System.out.println("Podana liczba: " + value);
5
6       try {
7         InputStream is = new FileInputStream("./abc.txt");
8         value = readInteger(is);
9       } catch (FileNotFoundException e) {
10        e.printStackTrace();
11      }
12
13    }
14
15  }
```

Zapis do strumienia

Listing: Example9.java

```
1  package examples;
2
3  import java.io.FileOutputStream;
4  import java.io.IOException;
5  import java.io.OutputStream;
6
7  public class Example9 {
8
9    public static void printString(String str, OutputStream stream) throws IOException {
10     stream.write(str.getBytes());
11   }
12
13   public static void main(String[] args) {
14     String str = "Hello!";
15     try {
16       printString(str, System.out);
17       FileOutputStream fStream = new FileOutputStream("./X.txt");
18       printString(str, fStream);
19     } catch (IOException e) {
20       e.printStackTrace();
21     }
22   }
```

Katalogi

Listing: Example12.java

```java
package examples;

import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class Example12 {
   public static void createDirectory(String examplePath)throws Exception {
      Path path = Paths.get(examplePath);
      Files.createDirectories(path);
   }

   public static boolean checkDirectoryExists(String examplePath) throws Exception {
      Path path = Paths.get(examplePath);
      return Files.exists(path);
   }

   public static void removeDirectory(String examplePath) throws Exception {
      Path path = Paths.get(examplePath);
      Files.delete(path);
   }

```

Katalogi

Listing: Example12Test.java

```java
package examples;

import static org.junit.Assert.*;

import org.junit.Test;


public class Example12Test {

   @Test
   public void testDirs() {
     String path = "./A/B/C";
     try {
       Example12.createDirectory(path);
       assertTrue("Directory Creation", Example12.checkDirectoryExists(path));
       Example12.removeDirectory(path);
     } catch (Exception e) {
       fail("An Exception has been caught:");
     }
   }
}
```

Tworzenie plików

Listing: Example13.java

```java
package examples;

import java.io.File;

public class Example13 {
   public static boolean createFile(String filename)throws Exception {
     File myFile = new File(filename);
     return myFile.createNewFile();
   }

   public static boolean checkFileExists(String filename)throws Exception {
     File myFile = new File(filename);
     return myFile.exists();
   }

   public static boolean removeFile(String filename)throws Exception {
     File myFile = new File(filename);
     return myFile.delete();
  }

}
```

Tworzenie plików

Listing: Example13Test.java

```java
package examples;

import static org.junit.Assert.*;

import org.junit.Test;

public class Example13Test {

   @Test
   public void testFiles() {
     String filename="zz.txt";
     try {
       assertTrue("Creation",Example13.createFile(filename));
       assertTrue("Exists?", Example13.checkFileExists(filename));
       assertTrue("Delete", Example13.removeFile(filename));
     } catch (Exception e) {
       fail();
     }
   }
}
```

Pliki tekstowe – zapis i odczyt

Listing: Example14.java

```
1  package examples;
2
3  import java.io.FileReader;
4  import java.io.FileWriter;
5
6  public class Example14 {
7     public static void write2File(String filename, String message)throws Exception {
8       FileWriter fileW = new FileWriter(filename);
9       fileW.write(message);
10      fileW.close();
11    }
12    public static String readFromFile(String filename)throws Exception {
13      FileReader fileR = new FileReader(filename);
14      StringBuffer buff = new StringBuffer();
15      int c;
16      while( (c = fileR.read()) != -1)
17        buff.append((char)c);
18      fileR.close();
19      return buff.toString();
20    }
21
22 }
```

Pliki tekstowe – zapis i odczyt

Listing: Example14Test.java

```java
package examples;

import static org.junit.Assert.*;

import org.junit.Test;

public class Example14Test {

   @Test
   public void testWR() {
     String filename="RWFile.txt";
     String message="Hello!";
     try {
       Example14.write2File(filename, message);
       assertTrue("ReadString", message.equals(Example14.readFromFile(filename)));
       Example13.removeFile(filename);
     } catch (Exception e) {
       fail();
     }
   }
}
```

Pliki o dostępie swobodnym

Listing: Example15.java

```
1  package examples;
2
3  import java.io.RandomAccessFile;
4
5  public class Example15 {
6     public static byte[] readFromFile(String filename, int position, int size)throws Exception {
7       RandomAccessFile file =new RandomAccessFile(filename, "r");
8       file.seek(position);
9       byte[] data = new byte[size];
10      file.read(data);
11      file.close();
12      return data;
13    }
14
15    public static void write2File(String filename, byte[] data, int position) throws Exception {
16      RandomAccessFile file =new RandomAccessFile(filename, "rw");
17      file.seek(position);
18      file.write(data);
19      file.close();
20    }
21 }
```

Pliki o dostępie swobodnym

Listing: Example15Test.java

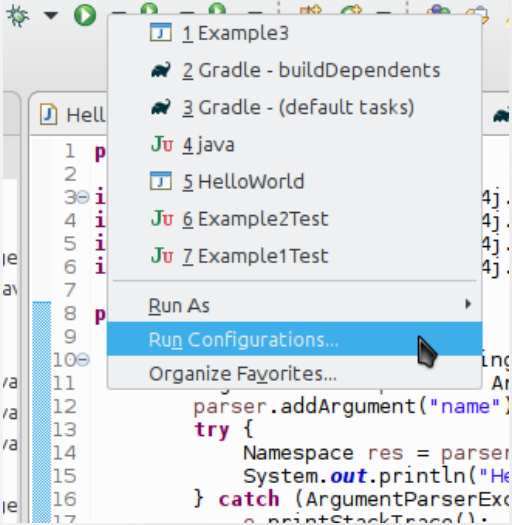
```
1  package examples;
2
3  import static org.junit.Assert.assertTrue;
4  import static org.junit.Assert.fail;
5
6  import java.util.Arrays;
7  import org.junit.Test;
8
9  public class Example15Test {
10    @Test
11    public void testRandomAccesFiles() {
12      byte[] bytes = {'a','c','d','e','f',};
13      int position=10;
14      String filename = "rndTestFile";
15      try {
16        Example15.write2File(filename, bytes, position);
17        byte[] rbytes = Example15.readFromFile(filename, position, bytes.length);
18        assertTrue("Equal arrays", Arrays.equals(bytes, rbytes));
19        Example13.removeFile(filename);
20      } catch (Exception e) {
21        fail();
22      }
```

Parsowanie argumentów

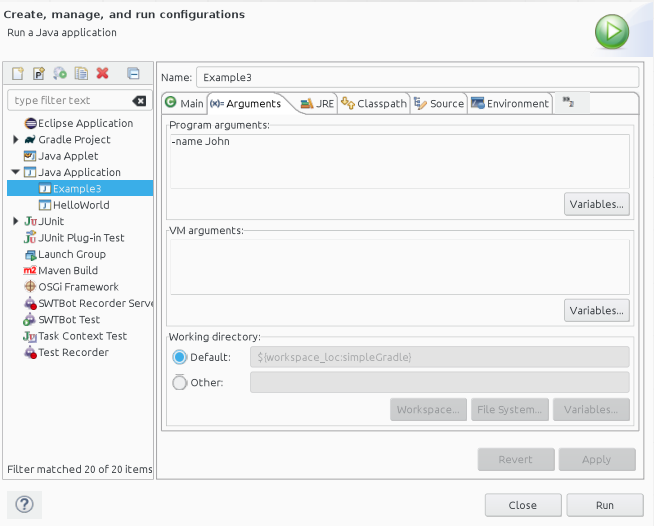
Listing: Example3.java

```java
package examples;
import java.util.List;
import net.sourceforge.argparse4j.ArgumentParsers;
import net.sourceforge.argparse4j.inf.ArgumentParser;
import net.sourceforge.argparse4j.inf.ArgumentParserException;
import net.sourceforge.argparse4j.inf.Namespace;
public class Example3 {
  public static void main(String[] args) {
    ArgumentParser parser = ArgumentParsers.newFor("Hello").build().description("Says Hallo");
    parser.addArgument("-name").type(String.class).nargs(1).help("Your name");
    try {
      Namespace res = parser.parseArgs(args);
      String yourName = (String)((List)res.getAttrs().get("name")).get(0);
      System.out.println("Hello " + yourName + "!");
    } catch (ArgumentParserException e) {
      e.printStackTrace();
    }
  }
}
```

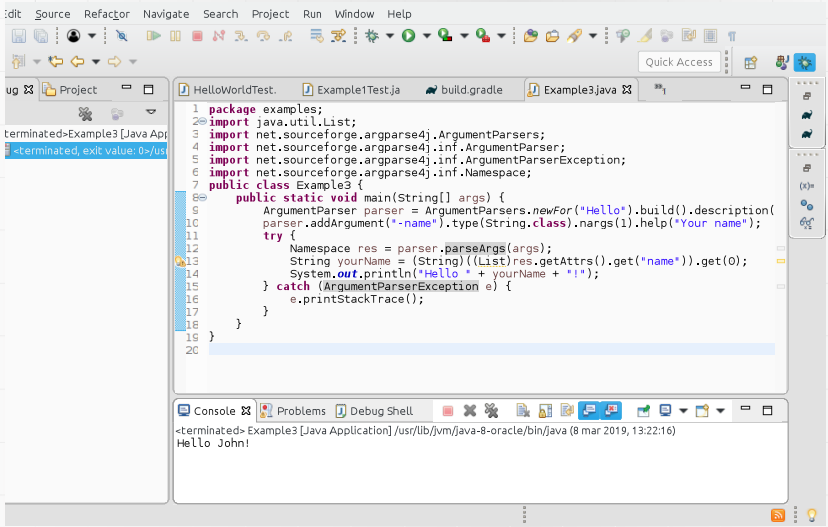
Parsowanie argumentów



Parsowanie argumentów



Parsowanie argumentów



Java i Eclipse – krótkie wprowadzenie

dr inż. Paweł Trajdos

Politechnika Wrocławska, Katedra Systemów i Sieci Komputerowych
Wyb. Wyspianskiego 27, 50-370 Wroclaw

5 lutego 2023