Programowanie i projektowanie obiektowe

Wyjątki, typy generyczne
Standardowa biblioteka klas

Paweł Daniluk

Wydział Fizyki

Jesień 2012

Wyjątki – rzucanie

Może się zdarzyć, że w czasie wykonywania programu występuje okoliczność nieprzewidziana (np. błąd) i trzeba ją jakoś obsłużyć.

W Javie do tego celu służą wyjątki.

```
void methodThrowingExceptions(Object obj) {
    if (obj == null) {
        // Throws exception of NullPointerException type
        throw new NullPointerException();
    }
    // Will not be called if obj was null
    doSomethingWithObject(obj);
}
```

Wyjątki – przechwytywanie

Wyjątki można przechwytywać i obsługiwać z dala od miejsca ich wystąpienia.

Nie jest możliwy powrót do miejsca wystąpienia wyjątku.

```
try {
    // Statements which may throw exceptions
    methodThrowingExceptions();
} catch (Exception ex) {
    // Exception caught and handled here
    reportException(ex);
} finally {
    // Statements always executed after the try/catch blocks
    freeResources();
}
```

Wyjątki – własne

Można definiować własne wyjątki dziedziczące z klasy `Exception` lub jej podklas.

Wszystkie wyjątki nie należące do klas `Error` i `RuntimeException` rzucane przez metodę muszą być zadeklarowane klauzulą `throws`

```
static class PegEmptyException extends Exception {};

int pop() throws PegEmptyException {
    if (top < 0) {
        throw new PegEmptyException();
    }
    return S[top--];
}
```

Wyjątki c.d.

W instrukcji `try...catch` można przechwytywać wiele wyjątków.

```
try {
    A.push(i);
} catch (DiskTooBigException ex) {
    ...
} catch (DiskAlreadyThereException ex) {
    ...
} catch (WrongDiskSizeException ex) {
    ...
}
```

Asercje

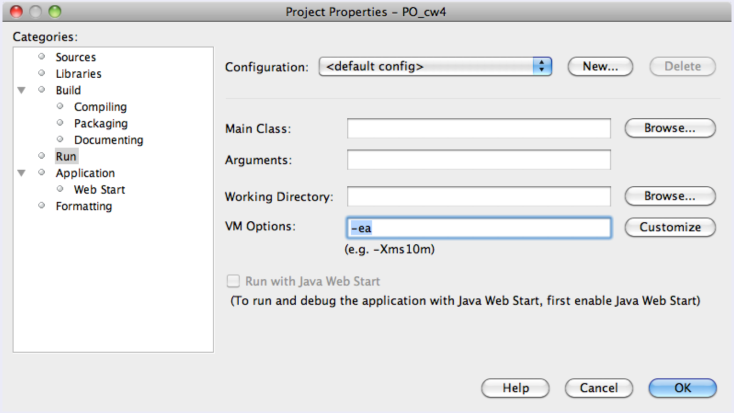
Często podczas szukania błędów opłaca się przerywać program, gdy pewien warunek logiczny nie jest spełniony (np. niezmiennik pętli).

```
// If n equals 0, AssertionError is thrown
assert n != 0;
/* If n equals 0, AssertionError will be thrown
with the message after the colon */
assert n != 0 : "n was equal to zero";
```

Asercje c.d.

Obsługę asercji trzeba włączyć parametrem -ea w maszynie wirtualnej Javy

W NetBeans



Project Properties – PO_cw4

Categories:
- Sources
- Libraries
- Build
  - Compiling
  - Packaging
  - Documenting
- Run
- Application
  - Web Start
- Formatting

Configuration: <default config>

Main Class:

Arguments:

Working Directory:

VM Options: -ea

(e.g. -Xms10m)

Run with Java Web Start

(To run and debug the application with Java Web Start, first enable Java Web Start)

Typy generyczne

Dzięki polimorfizmowi możemy mieć kontenery zawierające dowolne obiekty.

```
List v = new ArrayList();
v.add("test");
Integer i = (Integer)v.get(0);        // Run time error
```

Gdybyśmy umieli powiedzieć, że `v` będzie przechowywać wyłącznie ciągi znaków, wykrylibyśmy problem już podczas kompilacji.

```
List<String> v = new ArrayList<String>();
v.add("test");
Integer i = v.get(0); // (type error)  Compile time error
```

Typy generyczne c.d.

```
public interface List<E> {
    void add(E x);
    Iterator<E> iterator();
}
public interface Iterator<E> {
    E next();
    boolean hasNext();
}
```

Klasa generyczna

Definicja

```
/* This class has two type variables, T and V. T must be
a subtype of ArrayList and implement Formattable interface */

public class Mapper<T extends ArrayList & Formattable, V> {
    public void add(T array, V item) {
        // array has add method because it is an ArrayList subclass
        array.add(item);
    }
}
```

Zastosowanie

```
/* Mapper is created for CustomList as T and Integer as V.
CustomList must be a subclass of ArrayList and implement Formattable */

Mapper<CustomList, Integer> mapper = new Mapper<CustomList, Integer>();
```

Klasa generyczna c.d.

```
/* Any Mapper instance with CustomList as the first parameter
may be used regardless of the second one.*/

Mapper<CustomList, ?> mapper;
mapper = new Mapper<CustomList, Boolean>();
mapper = new Mapper<CustomList, Integer>();
```

```
/* Will not accept types that use anything but
a subclass of Number as the second parameter */

void addMapper(Mapper<?, ? extends Number> mapper) {
}
```

Generyczne metody

```
class Mapper {
    // The class itself is not generic, the constructor is
    <T, V> Mapper(T array, V item) {
    }
}

/* This method will accept only arrays of the same type as
the searched item type or its subtype*/
static <T, V extends T> boolean contains(T item, V[] arr) {
    for (T currentItem : arr) {
        if (item.equals(currentItem)) {
            return true;
        }
    }
    return false;
}
```

Generyczne intefejsy

```
interface Expandable<T extends Number> {
    void addItem(T item);
}

// This class is parametrized
class Array<T extends Number> implements Expandable<T> {
    void addItem(T item) {
    }
}

// And this is not and uses an explicit type instead
class IntegerArray implements Expandable<Integer> {
    void addItem(Integer item) {
    }
}
```

Java^{TM} Platform, Standard Edition 6

Pakiety

java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework

Java^{TM} Platform, Standard Edition 6

Pakiety

java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing beans – components based on the JavaBeansTM architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.instrument	Provides services that allow Java programming language agents to instrument programs running on the JVM.
java.lang.management	Provides the management interface for monitoring and

Java^{TM} Platform, Standard Edition 6

Pakiety

java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing beans – components based on the JavaBeansTM architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.instrument	Provides services that allow Java programming language agents to instrument programs running on the JVM.
java.lang.management	Provides the management interface for monitoring and management of the Java virtual machine as well as the operating system on which the Java virtual machine is running.
java.lang.ref	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects.
java.math	Provides classes for performing arbitrary-precision integer arithmetic (BigInteger) and arbitrary-precision decimal arithmetic (BigDecimal).
java.net	Provides the classes for implementing networking applications.
java.nio	Defines buffers, which are containers for data, and provides an overview of the other NIO packages.
java.nio.channels	Defines channels, which represent connections to entities that are capable of performing I/O operations, such as files and sockets; defines selectors, for multiplexed, non-blocking I/O operations.
java.nio.channels.spi	Service-provider classes for the java.nio.channels package.
java.nio.charset	Defines charsets, decoders, and encoders, for translating between bytes and Unicode characters.
java.nio.charset.spi	Service-provider classes for the java.nio.charset package.
java.rmi	Provides the RMI package.
java.rmi.activation	Provides support for RMI Object Activation.
java.rmi.dgc	Provides classes and interface for RMI distributed garbage-collection (DGC).
java.rmi.registry	Provides a class and two interfaces for the RMI registry.
java.rmi.server	Provides classes and interfaces for supporting the server side of RMI.
java.security	Provides the classes and interfaces for the security framework.
java.security.acl	The classes and interfaces in this package have been superseded by classes in the java.security package.
java.security.cert	Provides classes and interfaces for parsing and managing certificates, certificate revocation lists (CRLs), and certification paths.
java.security.interfaces	Provides interfaces for generating RSA (Rivest, Shamir and Adleman AsymmetricCipher algorithm) keys as defined in the RSA Laboratory Technical Note PKCS#1.

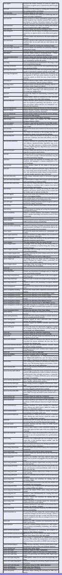
Java^{TM} Platform, Standard Edition 6

Pakiety

java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing beans – components based on the JavaBeansTM architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.instrument	Provides services that allow Java programming language agents to instrument programs running on the JVM.
java.lang.management	Provides the management interface for monitoring and management of the Java virtual machine as well as the operating system on which the Java virtual machine is running.
java.lang.ref	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects.
java.math	Provides classes for performing arbitrary-precision integer arithmetic (BigInteger) and arbitrary-precision decimal arithmetic (BigDecimal).
java.net	Provides the classes for implementing networking applications.
java.nio	Defines buffers, which are containers for data, and provides an overview of the other NIO packages.
java.nio.channels	Defines channels, which represent connections to entities that are capable of performing I/O operations, such as files and sockets; defines selectors, for multiplexed, non-blocking I/O operations.
java.nio.channels.spi	Service-provider classes for the java.nio.channels package.
java.nio.charset	Defines charsets, decoders, and encoders, for translating between bytes and Unicode characters.
java.nio.charset.spi	Service-provider classes for the java.nio.charset package.
java.rmi	Provides the RMI package.
java.rmi.activation	Provides support for RMI Object Activation.
java.rmi.dgc	Provides classes and interface for RMI distributed garbage-collection (DGC).
java.rmi.registry	Provides a class and two interfaces for the RMI registry.
java.rmi.server	Provides classes and interfaces for supporting the server side of RMI.
java.security	Provides the classes and interfaces for the security framework.
java.security.acl	The classes and interfaces in this package have been superseded by classes in the java.security package.
java.security.cert	Provides classes and interfaces for parsing and managing certificates, certificate revocation lists (CRLs), and certification paths.
java.security.interfaces	Provides interfaces for generating RSA (Rivest, Shamir and Adleman AsymmetricCipher algorithm) keys as defined in the RSA Laboratory Technical Note PKCS#1, and DSA (Digital Signature Algorithm) keys as defined in NIST's FIPS-186.
java.security.spec	Provides classes and interfaces for key specifications and algorithm parameter specifications.
java.sql	Provides the API for accessing and processing data stored in a data source (usually a relational database) using the JavaTM programming language.
java.text	Provides classes and interfaces for handling text, dates, numbers, and messages in a manner independent of natural languages.
java.text.spi	Service provider classes for the classes in the java.text package.
java.util	Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).
java.util.concurrent	Utility classes commonly useful in concurrent programming.
java.util.concurrent.atomic	A small toolkit of classes that support lock-free thread-safe programming on single variables.
java.util.concurrent.locks	Interfaces and classes providing a framework for locking and waiting for conditions that is distinct from built-in synchronization and monitors.
java.util.jar	Provides classes for reading and writing the JAR (Java ARchive) file format, which is based on the standard ZIP file format with an optional manifest file.
java.util.logging	Provides the classes and interfaces of the JavaTM 2 platform's core logging facilities.
java.util.prefs	This package allows applications to store and retrieve user and system preference and configuration data.
java.util.regex	Classes for matching character sequences against patterns specified by regular expressions.
java.util.spi	Service provider classes for the classes in the java.util package.
java.util.zip	Provides classes for reading and writing the standard ZIP and GZIP file formats.
javax.accessibility	Defines a contract between user-interface components and an assistive technology that provides access to those components.
javax.activation	
javax.activity	Contains Activity service related exceptions thrown by the ORB machinery during unmarshalling.
javax.annotation	
javax.annotation.processing	Facilities for declaring annotation processors and for allowing annotation processors to communicate with an annotation processing tool environment.
javax.crypto	Provides the classes and interfaces for cryptographic operations.
javax.crypto.interfaces	Provides interfaces for Diffie-Hellman keys as defined in RSA Laboratories' PKCS #3.
javax.crypto.spec	Provides classes and interfaces for key specifications and algorithm parameter specifications.
javax.imageio	The main package of the Java Image I/O API.
javax.imageio.event	A package of the Java Image I/O API dealing with synchronous notification of events during the reading and writing of images.
javax.imageio.metadata	A package of the Java Image I/O API dealing with reading and writing metadata.
javax.imageio.plugins.bmp	Package containing the public classes used by the built-in BMP plug-in.
javax.imageio.plugins.jpeg	Classes supporting the built-in JPEG plug-in.
javax.imageio.spi	A package of the Java Image I/O API containing the plug-in interfaces for readers, writers, transcoders, and streams, and a runtime registry.
javax.imageio.stream	A package of the Java Image I/O API dealing with low-level I/O from files and streams.
javax.jws	
javax.jws.soap	
javax.lang.model	Classes and hierarchies of packages used to model the Java programming language.
javax.lang.model.element	Interfaces used to model elements of the Java programming language.
javax.lang.model.type	Interfaces used to model Java programming language types.
javax.lang.model.util	Utilities to assist in the processing of program elements and types.
javax.management	Provides the core classes for the Java Management Exten-

Java^{TM} Platform, Standard Edition 6

Pakiety


[illegible]

Java^{TM} Platform, Standard Edition 6

Pakiety

Java^{TM} Platform, Standard Edition 6

Istotne pakiety

java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.math	Provides classes for performing arbitrary-precision integer arithmetic (BigInteger) and arbitrary-precision decimal arithmetic (BigDecimal).
java.net	Provides the classes for implementing networking applications.
java.sql	Provides the API for accessing and processing data stored in a data source (usually a relational database) using the JavaTM programming language.
java.text	Provides classes and interfaces for handling text, dates, numbers, and messages in a manner independent of natural languages.

JavaDoc

http://docs.oracle.com/javase/6/docs/api/

Opakowania dla typów podstawowych

Boolean	The Boolean class wraps a value of the primitive type boolean in an object.
Byte	The Byte class wraps a value of primitive type byte in an object.
Character	The Character class wraps a value of the primitive type char in an object.
Double	The Double class wraps a value of the primitive type double in an object.
Float	The Float class wraps a value of primitive type float in an object.
Integer	The Integer class wraps a value of the primitive type int in an object.
Long	The Long class wraps a value of the primitive type long in an object.
Number	The abstract class Number is the superclass of classes BigDecimal, BigInteger, Byte, Double, Float, Integer, Long, and Short.
Short	The Short class wraps a value of primitive type short in an object.

Automatyczne pudełkowanie

```
int foo = 42; // Primitive type
Integer bar = foo; /* foo is boxed to bar, bar is of Integer ty
                   which serves as a wrapper for int */
int foo2 = bar; // Unboxed back to primitive type
```

Klasa Integer

Atrybuty

static int	MAX_VALUE	A constant holding the maximum value an int can have, $2^{31}-1$.
static int	MIN_VALUE	A constant holding the minimum value an int can have, -2^{31}.
static int	SIZE	The number of bits used to represent an int value in two's complement binary form.

Konstruktory

Integer(int value)	Constructs a newly allocated Integer object that represents the specified int value.
Integer(String s)	Constructs a newly allocated Integer object that represents the int value indicated by the String parameter.

Klasa Integer c.d.

Wybrane metody

static int	bitCount(int i)	Returns the number of one-bits in the two's complement binary representation of the specified int value.
byte	byteValue()	Returns the value of this Integer as a byte.
int	compareTo(Integer anotherInteger)	Compares two Integer objects numerically.
double	doubleValue()	Returns the value of this Integer as a double.
float	floatValue()	Returns the value of this Integer as a float.
int	intValue()	Returns the value of this Integer as an int.
long	longValue()	Returns the value of this Integer as a long.
static int	lowestOneBit(int i)	Returns an int value with at most a single one-bit, in the position of the lowest-order (żightmost") one-bit in the specified int value.
static int	numberOfLeadingZeros(int i)	Returns the number of zero bits preceding the highest-order (łeftmost") one-bit in the two's complement binary representation of the specified int value.
static int	numberOfTrailingZeros(int i)	Returns the number of zero bits following the lowest-order (żightmost") one-bit in the two's complement binary representation of the specified int value.
static int	parseInt(String s)	Parses the string argument as a signed decimal integer.
static int	signum(int i)	Returns the signum function of the specified int value.
static String	toBinaryString(int i)	Returns a string representation of the integer argument as an unsigned integer in base 2.
static String	toHexString(int i)	Returns a string representation of the integer argument as an unsigned integer in base 16.
static String	toOctalString(int i)	Returns a string representation of the integer argument as an unsigned integer in base 8.
String	toString()	Returns a String object representing this Integer's value.
static String	toString(int i)	Returns a String object representing the specified integer.
static Integer	valueOf(int i)	Returns a Integer instance representing the specified int value.
static Integer	valueOf(String s)	Returns an Integer object holding the value of the specified String.

Klasa String

Wybrane metody

char	charAt(int index)	Returns the char value at the specified index.
int	compareTo(String anotherString)	Compares two strings lexicographically.
int	compareToIgnoreCase(String str)	Compares two strings lexicographically, ignoring case differences.
String	concat(String str)	Concatenates the specified string to the end of this string.
boolean	contains(CharSequence s)	Returns true if and only if this string contains the specified sequence of char values.
boolean	endsWith(String suffix)	Tests if this string ends with the specified suffix.
static String	format(String format, Object... args)	Returns a formatted string using the specified format string and arguments.
int	indexOf(String str)	Returns the index within this string of the first occurrence of the specified substring.
boolean	isEmpty()	Returns true if, and only if, length() is 0.
int	lastIndexOf(String str)	Returns the index within this string of the rightmost occurrence of the specified substring.
int	length()	Returns the length of this string.
String	replace(char oldChar, char newChar)	Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
String[]	split(String regex)	Splits this string around matches of the given regular expression.
boolean	startsWith(String prefix)	Tests if this string starts with the specified prefix.
String	substring(int beginIndex, int endIndex)	Returns a new string that is a substring of this string.
char[]	toCharArray()	Converts this string to a new character array.
String	toLowerCase()	Converts all of the characters in this String to lower case using the rules of the default locale.
String	trim()	Returns a copy of the string, with leading and trailing whitespace omitted.
static String	valueOf(Object obj)	Returns the string representation of the Object argument.

Operacje na plikach – znak po znaku

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class CopyCharacters {
    public static void main(String[] args) throws IOException {
        FileReader inputStream = null;
        FileWriter outputStream = null;

        try {
            inputStream = new FileReader("xanadu.txt");
            outputStream = new FileWriter("characteroutput.txt");

            int c;
            while ((c = inputStream.read()) != -1) {
                outputStream.write(c);
            }
        } finally {
            if (inputStream != null) {
                inputStream.close();
            }
            if (outputStream != null) {
                outputStream.close();
            }
        }
    }
}
```

Operacje na plikach – linia po linii

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.IOException;

public class CopyLines {
    public static void main(String[] args) throws IOException {
        BufferedReader inputStream = null;
        PrintWriter outputStream = null;

        try {
            inputStream =
                new BufferedReader(new FileReader("xanadu.txt"));
            outputStream =
                new PrintWriter(new FileWriter("characteroutput.txt"));

            String l;
            while ((l = inputStream.readLine()) != null) {
                outputStream.println(l);
            }
        } finally {
            if (inputStream != null) {
                inputStream.close();
            }
            if (outputStream != null) {
                outputStream.close();
            }
        }
    }
}
```

Operacje na plikach – wejście formatowane

```
import java.io.*;
import java.util.Scanner;

public class ScanXan {
    public static void main(String[] args) throws IOException {
        Scanner s = null;
        try {
            s = new Scanner(new BufferedReader(new FileReader("xanadu.txt")));

            while (s.hasNext()) {
                System.out.println(s.next());
            }
        } finally {
            if (s != null) {
                s.close();
            }
        }
    }
}
```

Operacje na plikach – wejście formatowane c.d.

```
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;
import java.util.Scanner;
import java.util.Locale;

public class ScanSum {
    public static void main(String[] args) throws IOException {
        Scanner s = null;
        double sum = 0;
        try {
            s = new Scanner(
                    new BufferedReader(new FileReader("usnumbers.txt")));
            s.useLocale(Locale.US);


            while (s.hasNext()) {
                if (s.hasNextDouble()) {
                        sum += s.nextDouble();
                    } else {
                        s.next();
                    }
            }
        } finally {
            s.close();
        }

        System.out.println(sum);
    }
}
```

Operacje na plikach – wyjście formatowane

```
public class Root {
    public static void main(String[] args) {
        int i = 2;
        double r = Math.sqrt(i);

        System.out.print("The square root of ");
        System.out.print(i);
        System.out.print(" is ");
        System.out.print(r);
        System.out.println(".");

        i = 5;
        r = Math.sqrt(i);
        System.out.println("The square root of " + i + " is " + r + ".");
    }
}
```

```
public class Root2 {
    public static void main(String[] args) {
        int i = 2;
        double r = Math.sqrt(i);

        System.out.format("The square root of %d is %f.%n", i, r);
    }
}
```

Standardowe wejście/wyjście

Klasa System

static PrintStream	err	The “standard” error output stream.
static InputStream	in	The “standard” input stream.
static PrintStream	out	The “standard” output stream.

Zadanie 1 – Testy

Zadanie

Tworząc odpowiednie przykłady przetestuj działanie mechanizmów przedstawionych na wykładzie.

Wskazówka

Konstrukcje niepoprawne (np. próby dostępu do atrybutów prywatnych) po przetestowaniu zasłoń komentarzem.

Zadanie 2 – Uliniowienia sekwencji

Zadanie

Uzupełnij program obliczający uliniowienia algorytmem Smitha-Watermana o wczytywanie sekwencji z pliku i wypisywanie uliniowienia do pliku.

Zadanie 3 – Figury geometryczne MkII

Zadanie

Utwórz hierarchię klas służącą do przechowywania informacji o figurach geometrycznych (kwadrat, prostokąt, koło, trójkąt) pozwalającą na wykonywanie następujących operacji (tam gdzie to możliwe):

- obliczanie obwodu i pola,
- obliczanie długości najdłuższego boku,
- obliczanie promienia okręgu opisanego na figurze,
- wypisywanie informacji.

Figury znają swoje położenie w układzie współrzędnych.
Zaimplementuj klasę pozwalającą na przechowywanie figur (kolekcję figur) oraz podklasę tej kolekcji, do której nie można dodać figury jeżeli przecina się ona z figurą już będącą w kolekcji. Wykorzystaj wyjątki do obsługi błędów.

Zadanie 3 – Figury geometryczne MkII c.d.

Wskazówki

- Figury muszą umieć sprawdzić czy się przecinają.
- Każdą figurę da się wpisać w prostokąt o bokach równoległych do osi układu współrzędnych. Jeżeli takie prostokąty nie przecinają się, to figury tym bardziej.
- Drzewa binarnego wyszukiwania są wygodne do przechowywania porządkowania przedziałów. Przydadzą się dwa, dla obydwu kierunków w układzie współrzędnych. Więcej inspiracji w rozdziale 15.3 Wprowadzenia do algorytmów.