

1 Klasa File

Klasa File zapewnia podstawowe operacje na plikach

```
package com.adeik.javatest.IO;

import java.io.File;
import java.io.IOException;

public class FileTest {

    public static void main(String[] args){
        File f = new File("/home/adam/abc.txt");
        try {
            f.createNewFile();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2 Writer

Klasy dziedziczące po klasie Writer realizują zapis danych

```
package com.adeik.javatest.IO;

import java.io.FileWriter;
import java.io.IOException;

public class FileWriterTest {

    public static void main(String[] args){
        try {
            FileWriter fw = new FileWriter("/home/adam/abc.txt");
            fw.write("Hello ");
            fw.write("World");
            fw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

3 Reader

Klasy dziedziczące po klasie Reader realizują odczyt danych

3.1 FileReader

```
package com.adeik.javatest.IO;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class FileReaderTest {

    public static void main(String[] args) {
        int i;
        try {
            FileReader fr = new FileReader("/home/adam/abc.txt");
            while ((i = fr.read()) != -1)
                System.out.print((char)i);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e){
            e.printStackTrace();
        }
    }
}
```

3.2 BufferedReader

```
package com.adeik.javatest.IO;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class BufferedReaderTest {

    public static void main(String[] args){
        String s;
        try {
            BufferedReader br = new BufferedReader(new FileReader("/home/adam/abc.txt"));
            while ((s = br.readLine()) != null){
                System.out.println(s);
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e){
            e.printStackTrace();
        }
    }
}
```

```
}
```

4 Strumienie

Strumienie pozwalają na zapisywanie ciągów do plików, tablic bajtów itp.

```
package com.adeik.javatest.IO;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;

public class StreamTest {

    public static void main(String[] args){
        int i;
        try {
            ByteArrayOutputStream os = new ByteArrayOutputStream();
            os.write(1);
            os.write(2);
            os.write("Hello".getBytes());
            byte[] bytes = os.toByteArray();
            ByteArrayInputStream is = new ByteArrayInputStream(bytes);
            while ((i = is.read()) != -1)
                System.out.print(i);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

5 Serializacja obiektów

W celu zapisywania obiektów należy wykorzystać interfejs `Serializable`. Obiekt wykorzystujący serializację można zapisać do pliku lub przesłać przez gniazdko sieciowe.

```
package com.adeik.javatest.IO;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

class SerializableClass implements Serializable{
    int i;
    boolean b;
    String s;

    public SerializableClass(int i, boolean b, String s){
```

```

    this.i = i;
    this.b = b;
    this.s = s;
}

@Override
public String toString(){
    return "Obiekt klasy SerializableClass "+i+", "+b+", "+s;
}
}

public class SerializationTest{

    public static void main(String[] args){
        FileOutputStream fos = null;
        FileInputStream fis = null;
        try {
            fos = new FileOutputStream("/home/adam/abc.txt");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            SerializableClass so = new SerializableClass(0, true, "abcd");
            oos.writeObject(so);
            so = new SerializableClass(6, false, "xyz");
            oos.writeObject(so);
            oos.close();

            fis = new FileInputStream("/home/adam/abc.txt");
            ObjectInputStream ois = new ObjectInputStream(fis);
            so = (SerializableClass)ois.readObject();
            System.out.println(so);
            so = (SerializableClass)ois.readObject();
            System.out.println(so);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

6 Przekierowanie wejścia i wyjścia

Standardowe strumienie mogą zostać przekierowane do pliku:

- `System.setOut(PrintStream);`
- `System.setIn(InputStream);`
- `System.setErr(PrintStream);`

7 Zadania do wykonania

1. Zapoznać się z dokumentacją klasy `File`

2. Zapoznać się z dokumentacją klas wraz z ich klasami pochodnymi: Writer, Reader, InputStream, OutputStream
3. Zapisać i odczytać z pliku wartości typów prostych
4. Zapisać i odczytać z pliku obiekty przygotowanej klasy
5. Przekierować standardowe wejście i wyjście do plików