Using Java Platform API Library Classes and Documentation

Zbigniew Dendzik
Institute of Physics, University of Silesia

2025

Introduction

The Java platform consists of two components – the virtual machine (bytecode interpreter) and a library of classes and interfaces. In today's example we will analyse ways to use library classes that are part of any complete implementation of the Java platform. Today's exercise will also involve developing the necessary skills in using documentation (Java Platform API Specification). As in other examples, if a library class is used in an exercise, its first occurrence will at the same time be a link to its documentation.

Example I

Below you will find the source code of a program that creates an object representing a rectangle with a given vertex, width and height, and performs on it a translation operation by the vector $(1, -1)$. Analyse the example and compare the use of the library class `java.awt.Rectangle` with the use of your own implementation of the `Rectangle` class from previous classes. Recall the issues of class naming and defining and importing packages (lecture) and analyse them in the context of using Java Platform API classes.

```
1  import java.awt.Rectangle;
2
3  public class Program
4  {
5     public static void main(String[] args)
6     {
7        Rectangle obj=new Rectangle(0,0,4,3);
8
9        obj.translate(1,-1);
10
11       System.out.println(obj);
12    }
13 }
```

Listing 1: Example I: using java.awt.Rectangle class

Ex. 3.1

Familiarise yourself with the documentation of the `java.awt.Rectangle` library class. Based on information from previous exercises, identify the fields, constructors and methods of this class and answer the question of what their interpretation is and what they are used for.

Ex. 3.2

Create an object `obj1` representing a rectangle with a vertex at point $(0, 0)$, width 4 and height 3, and an object `obj2` representing a rectangle with a vertex at point $(1, 1)$, width 4 and height 3. Using the documentation of the `java.awt.Rectangle` class, find the appropriate methods and write a program that will create an object representing a rectangle that is the intersection (common part) of `obj1` and `obj2`. Compile and test the example.

Ex. 3.3

Create an object `obj1` representing a rectangle with a vertex at point $(1, 1)$, width 4 and height 5, and an object `obj2` representing a rectangle with a vertex at point $(2, 0)$, width 2 and height 3. Using the documentation of the `java.awt.Rectangle` class, find the appropriate methods and write a program that will check whether the rectangle `obj1` is contained within the rectangle `obj2`. Compile and test the example.

Ex. 3.4

Create an object `obj` representing a rectangle with a vertex at point $(-3, 0)$, width 6 and height 3. Using the documentation of the `java.awt.Rectangle` class, find the appropriate methods and write a program that will check whether the point $(2, -1)$ lies inside the rectangle `obj`. Compile and test the example.

Ex. 3.5

Create an object `obj1` representing a rectangle with a vertex at point $(1, 1)$, width 4 and height 5, and an object `obj2` representing a rectangle with a vertex at point $(4, -3)$, width 4 and height 3. Using the documentation of the `java.awt.Rectangle` class, find the appropriate methods and write a program that will check whether the rectangle `obj1` intersects the rectangle `obj2`. Compile and test the example.

Example II

Below you will find an example of using Java Platform API classes for keyboard input. The example also illustrates the concept of exception handling and the related syntax.

```
1 import java.io.*;
2
3 public class A
4 {
5    static double RATE=3.8;
```

```
6
7     public static void main(String[] args)
8     {
9        try
10       {
11          BufferedReader br=new BufferedReader(new
               InputStreamReader(System.in));
12
13          System.out.print("$: ");
14          String str=br.readLine();
15
16          double x=Double.parseDouble(str);
17          System.out.println("PLN: "+x*RATE);
18       }
19
20       catch(IOException e1)
21       {
22          System.out.println("input/output exception");
23       }
24
25       catch(NumberFormatException e2)
26       {
27          System.out.println("invalid number format");
28       }
29    }
30 }
```

Listing 2: Example II: keyboard input

Ex. 3.6

Extend the example to allow selection between several different currencies and then conversion at a given exchange rate.

Example III

Below you will find an example of using Java Platform API classes for simple operations on text files. The example also illustrates the concept of exception handling and the related syntax.

```
1  import java.io.*;
2
3  public class A
4  {
5     public static void main(String[] args)
6     {
7        BufferedReader br=new BufferedReader(new InputStreamReader(
            System.in));
8
9        try
10       {
```

```
11          System.out.print("first name: ");
12          String firstName=br.readLine();
13          System.out.print("last name: ");
14          String lastName=br.readLine();
15
16          PrintWriter file1=new PrintWriter(new BufferedWriter(new
                FileWriter("file.txt",true)));
17          file1.println(firstName+" "+lastName);
18          file1.close();
19      }
20      catch(Exception e){System.out.println(e);}
21
22
23
24      System.out.println("\n-- from file --");
25
26      try
27      {
28          BufferedReader file2=new BufferedReader(new FileReader("
                file.txt"));
29          String str;
30
31          while(file2.ready())
32          {
33              str=file2.readLine();
34              System.out.println(str);
35          }
36
37          file2.close();
38      }
39      catch(Exception e){System.out.println(e);}
40  }
41 }
```

Listing 3: Example III: text file operations

Ex. 3.7

Write a `Rectangle` class equipped with a `void save(String file)` method that saves the contents of all fields of this rectangle to a text file with the given name.

Ex. 3.8

Write a program demonstrating the use of classes from any available EXTERNAL library (network, graphics, mathematical or any other applications). Below you will find an example of using an external open source library containing an implementation of encryption using the asymmetric RSA algorithm, which can be downloaded from `www.bouncycastle.org`. You can use this library or any other of your choice.

```
1 import java.io.*;
2 import java.security.*;
```

```
3  import javax.crypto.*;
4
5  public class RSATest
6  {
7     public static void main(String args[]) throws Exception
8     {
9        System.out.println("starting to register provider");
10       Provider prov = new org.bouncycastle.jce.provider.
            BouncyCastleProvider();
11       Security.addProvider(prov);
12
13       System.out.println("starting to generate key pair generator
            ");
14       KeyPairGenerator kpgen = KeyPairGenerator.getInstance("RSA"
            ,"BC");
15       kpgen.initialize(1024, new java.security.SecureRandom());
16
17       System.out.println("starting to generate key pair");
18       KeyPair pair = kpgen.genKeyPair();
19
20       System.out.println("starting to generate private key");
21       PrivateKey priv = pair.getPrivate();
22
23       System.out.println("starting to generate public key");
24       PublicKey pub = pair.getPublic();
25
26
27
28       //-- CipherStream: encryption --
29       System.out.println("starting to generate cipher");
30       javax.crypto.Cipher c1 = javax.crypto.Cipher.getInstance("
            RSA/ECB/PKCS1Padding","BC");
31       c1.init(Cipher.ENCRYPT_MODE,pub);
32
33       System.out.println("starting to set up output stream");
34       ObjectOutputStream out=new ObjectOutputStream(new
            CipherOutputStream(new FileOutputStream("file.txt"),c1));
35
36       System.out.println("starting to write object to stream");
37       out.writeObject(new String("greetings from auntie"));
38
39       out.flush();
40       out.close();
41
42
43
44       //-- CipherStream: decryption  --
45       System.out.println("starting to generate cipher");
46       javax.crypto.Cipher c2 = javax.crypto.Cipher.getInstance("
            RSA/ECB/PKCS1Padding","BC");
47       c2.init(Cipher.DECRYPT_MODE,priv);
```

```
48
49          System.out.println("starting to set up input stream");
50          ObjectInputStream in=new ObjectInputStream(new
               CipherInputStream(new FileInputStream("file.txt"),c2));
51
52          System.out.println("starting to read object from stream");
53          String str=(String)in.readObject();
54
55          System.out.println("\n-- "+str+" --\n");
56          in.close();
57      }
58  }
```

Listing 4: Example RSA encryption using BouncyCastle library