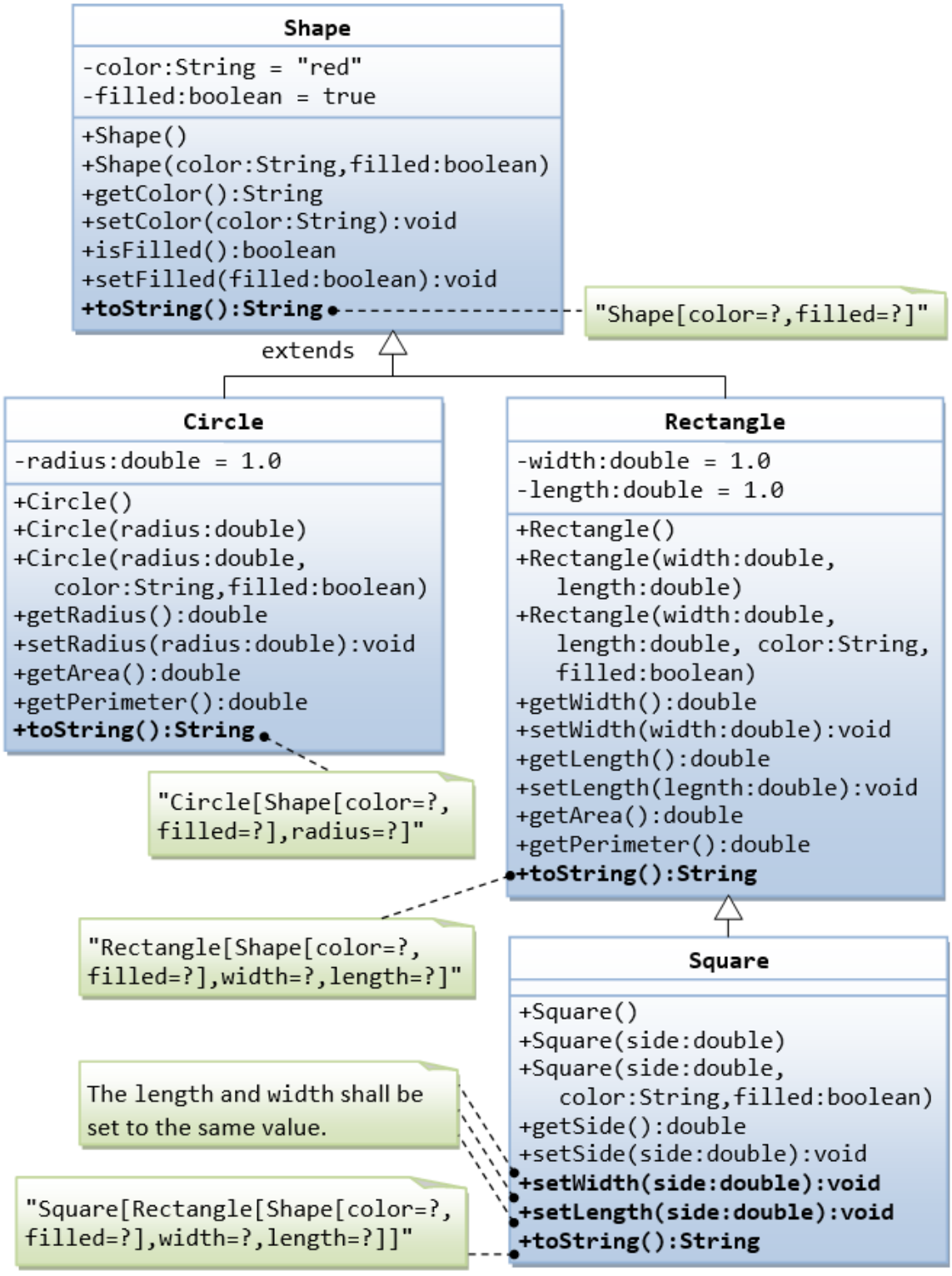
Assignment 6
March 7, 2020

Part I: Inheritance



Write a superclass called Shape (as shown in the class diagram), which contains:

- Two instance variables color (String) and filled (boolean).
- Two constructors: a no-arg (no-argument) constructor that initializes the color to "green" and filled to true, and a constructor that initializes the color and filled to the given values.
- Getter and setter for all the instance variables. By convention, the getter for a boolean variable xxx is called isXXX() (instead of getXxx() for all the other types).
- A toString() method that returns "A Shape with color of xxx and filled/Not filled".

Write a test program to test all the methods defined in Shape.

Write two subclasses of Shape called Circle and Rectangle, as shown in the class diagram.

The Circle class contains:

- An instance variable radius (double).
- Three constructors as shown. The no-arg constructor initializes the radius to 1.0.
- Getter and setter for the instance variable radius.
- Methods getArea() and getPerimeter().
- Override the toString() method inherited, to return "A Circle with radius=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.

The Rectangle class contains:

- Two instance variables width (double) and length (double).
- Three constructors as shown. The no-arg constructor initializes the width and length to 1.0.
- Getter and setter for all the instance variables.
- Methods getArea() and getPerimeter().
- Override the toString() method inherited, to return "A Rectangle with width=xxx and length=zzz, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.

Write a class called Square, as a subclass of Rectangle. Convince yourself that Square can be modeled as a subclass of Rectangle. Square has no instance variable, but inherits the instance variables width and length from its superclass Rectangle.

- Provide the appropriate constructors (as shown in the class diagram). Hint:

```
public Square(double side) {
   super(side, side);  // Call superclass Rectangle(double, double)
}
```

- Override the toString() method to return "A Square with side=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.
- Do you need to override the getArea() and getPerimeter()? Try them out.
- Override the setLength() and setWidth() to change both the width and length, so as to maintain the square geometry.

Part II: Exceptions

1. Take the following code, ListOfNumbers.java:

```
/*
 * Copyright (c) 1995, 2008, Oracle and/or its affiliates. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 *   - Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 *
 *   - Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the distribution.
 *
 *   - Neither the name of Oracle or the names of its
 *     contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
 * IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
import java.io.*;
import java.util.List;
import java.util.ArrayList;

public class ListOfNumbers {
    private List<Integer> list;
    private static final int SIZE = 10;

    public ListOfNumbers () {
        list = new ArrayList<Integer>(SIZE);
        for (int i = 0; i < SIZE; i++)
            list.add(new Integer(i));
    }
    public void writeList() {
        PrintWriter out = null;
        try {
            System.out.println("Entering try statement");
            out = new PrintWriter(new FileWriter("OutFile.txt"));
            for (int i = 0; i < SIZE; i++)
                out.println("Value at: " + i + " = " + list.get(i));
        } catch (IndexOutOfBoundsException e) {
            System.err.println("Caught IndexOutOfBoundsException: " +
                               e.getMessage());
        } catch (IOException e) {
            System.err.println("Caught IOException: " + e.getMessage());
        } finally {
            if (out != null) {
                System.out.println("Closing PrintWriter");
                out.close();
            } else {
                System.out.println("PrintWriter not open");
            }
        }
    }
}
```

Add a `readList` method to ListOfNumbers.java. This method should read in int values from a file, print each value, and append them to the end of the vector. You should catch all appropriate errors. You will also need a text file containing numbers to read in.

Modify the following cat method so that it will compile.

```
public static void cat(File file) {
    RandomAccessFile input = null;
    String line = null;

    try {
        input = new RandomAccessFile(file, "r");
        while ((line = input.readLine()) != null) {
            System.out.println(line);
        }
        return;
    } finally {
        if (input != null) {
            input.close();
        }
    }
}
```

2. Capture all exceptions in the following program, printing out error messages that describe the type of error that occurred.

```
import java.io.*;
public class Exercise2 {
      public static void main(String[] args) {
            int n=10;
            int[] v = new int[n];
            FileReader f = new FileReader("dati.txt");
            BufferedReader in = new BufferedReader(f);
            int i=0;
            String linea = in.readLine();
            while (linea!=null) {
                  v[i] = Integer.parseInt(linea);
                  linea = in.readLine();
                  i++;
            }
            f.close();
      }
}
```

3. Create a Java class Matrix to represent bidimensional matrices of real numbers. The class should export the following methods:

    • Matrix(int n, int m) : constructor that creates a matrix of size nxm, with all values initially set to 0;
    • void save(String filename) : that saves the content of the matrix on the file specified by filename;
    • static Matrix read(String filename) : that reads the data about a matrix from the file specified by filename, creates the matrix, and returns it;
    • Matrix sum(Matrix m) : that returns the matrix that is the sum of the object and of m, if the two matrices have the same dimensions, and null otherwise;
    • Matrix product(Matrix m) : that returns the matrix that is the product of the object and of m, if the two matrices have compatible dimensions, and null otherwise.

Define the exceptions that are necessary to catch the possible errors that can occur in the class Matrix:

• `ExceptionWrongMatrixValues` that is thrown in the method read() if the data on the file does not correspond to numeric values, or if the data are not consistent with the form of a matrix (e.g., the rows have different length);

• `ExceptionWrongMatrixDimension` that is thrown in the method read() if the data on the file do not correspond to the dimension of the matrix. Modify the class Matrix in such a way that it generates the new exceptions when necessary.