PR1 – Formular für Lesenotizen
WS2020/21

Nachname	Vorname	Matrikelnummer	Abgabedatum:
Abdel Kader	Schehat	1630110	26.11.2020

Programmlogik und indefinite Schleifen (L.5.4-L.5.6)
Dateien und Exceptions (L.6.6-L.6.7)

Lernzielfragen:

a) Setzen Sie break und continue in folgender Anwendung ein: Einlesen von ganzen Zahlen vom Benutzer, Aufsummieren aller durch 5 teilbaren Werte, Ende und Ausgabe der Summe bei Eingabe von -1

```
import java.util.Scanner;
public class Aufa {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int sum = 0;
        int input = 0;

        while (true) {
            System.out.print("Bitte Zahl eingeben (verlasse mit -1): ");
            input = scanner.nextInt();
            if (input == -1) {
                break;
            }
            if (input % 5 == 0) {
                sum += input;
            }
        }
        System.out.println("Summe: " + sum);
    }
}
```

e) Beschreiben Sie den Ablauf zur Laufzeit beim Erzeugen, Weiterreichen und Abfangen einer Exception. Skizzieren Sie ein Beispiel mit mindestens zwei beteiligten Methoden und unterscheiden die Fälle von überprüfungsbedürftigen und nicht überprüfungs-bedürftigen Exceptions.

- Bei der Erzeugung einer Exception kann sie entweder sofort behandelt werden mit **catch** oder weitergereicht werden mit **throws** wie bei überprüfungsbedürftige Exceptions.
- Nicht überprüfungsbedürftige Exceptions muss man nicht throwen und somit catchen, jedoch es sich solche Fehler zu behandeln

g) Schreiben Sie ein Programm, das den Benutzer nach einem Dateinamen und einer Textzeile fragt. Hängen Sie die Textzeile ans Ende der Datei an.

```
import java.io.*;
import java.util.Scanner;
public class Auff {
    public static void main(String[] args) throws FileNotFoundException{
        Scanner scanner = new Scanner(System.in);
        System.out.print("Dateinamen angeben: ");
        String name = scanner.nextLine() + ".txt";
        PrintStream output = new PrintStream(new FileOutputStream(new File(name), true));
        System.out.print("Geben Sie Inhalt an: ");
        String text = scanner.nextLine();
        output.println(text);
        output.close();
        scanner.close();
    }
}
```

h) Schreiben Sie ein Programm, das aus einer Datei paarweise zwei ganze Zahlen ausliest. Das Programm soll jeweils die erste durch die zweite Zahl dividieren (Ganzzahldivision) und das Ergebnis ausgeben. Division durch 0 sollen Sie mit try/catch abfangen. Schließen Sie die Datei auch im Fehlerfall.

```
import java.io.*;
import java.util.Scanner;
public class Aufg {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        Scanner input = null;
        String name = null;
        do {
            System.out.print("Geben Sie einen Dateinamen an: ");
            name = console.nextLine() + ".txt";
            try {
                input = new Scanner(new File(name));
            } catch (FileNotFoundException e) {
                System.out.println("Diesen Dateinamen gibt es nicht");
            }
        } while(input == null);
        int number1 = 0;
        int number2 = 0;
        int ergebnis = 0;
        while (input.hasNextLine()) {
            number1 = input.nextInt();
            number2 = input.nextInt();
            try {
                ergebnis = number1 / number2;
                System.out.println(number1 + " / " + number2 + " = " + ergebnis);
            } catch (ArithmeticException e) {
                System.out.println(number1 + " / " + number2 + ": Unerlaubte Division duch 0!");
            }
        }
        input.close();
        console.close();
    }
}
```

Notizen:

Token-basierte Verarbeitung von String

- Scanner <name> = new Scanner(<String>);

Abwechselndes Token- und Zeilenbasiertes Einlesen

- Möglichst vermeiden, da häufig Fehler auftreten, wenn man z.B.: bei der Konsole nach scanner.next() danach scanner.nextLine() abfragt, da nextLine() nur den Zeilenumbruch liest

try & catch & finally

Syntax:

```
try {
   <statement(s) >;
} catch (<exception-type> <name>) {
   <statement(s)>;
} catch (<exception-type> <name>) {
   <statement(s)>;
} finally {
   <statement(s)>;
}
```

<name> meist e

Scanner-Objekte vor dem try Block deklarieren & initialisieren mit Scanner <name> = null wegen Geltungsbereich

Ausgabe in Dateien

- Aus io package
- kann Methoden wie print oder println benutzten
- Wenn Datei nicht existiert wird neu angelegt bzw. überschrieben. Überprüfungsbedürftig wie Scanner mit FileNotFoundException. Datei nicht gleichzeitig lesen und schreiben => Fehler

```
PrintStream <name> =  new PrintStream(new File("<file name>"));
```

- true bedeutet, Datei wird nicht überschrieben

```
PrintStream output=
  new PrintStream(new FileOutputStream(new File("output.txt"),
                                       true));
```