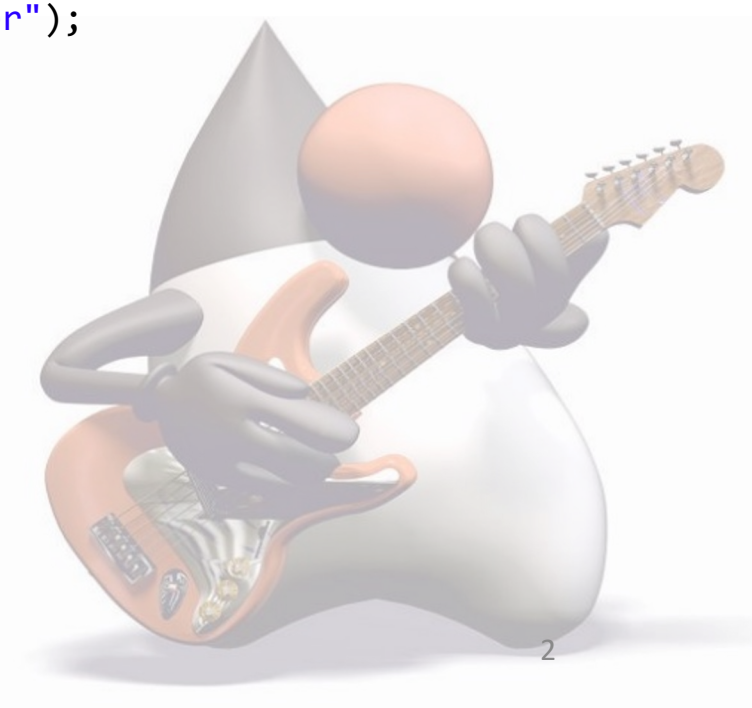
Java Fundamentals

Exception Handling

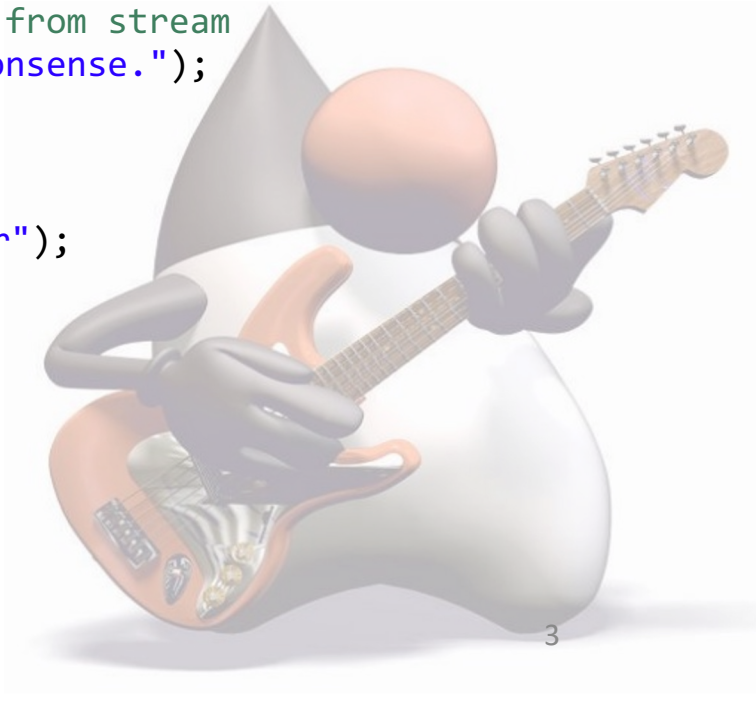
```java
public class GuessNumber {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		int guessedNumber = 0;
		int randomNumber = (int)((Math.random() * 100) + 1);
		System.out.println("Guess a number between 1 and 100");
		while (guessedNumber!=randomNumber){
			System.out.println("Your Guess:");
			guessedNumber = scanner.nextInt();
			if (guessedNumber>randomNumber){
				System.out.println("This number is too large.");
			} else if (guessedNumber<randomNumber){
				System.out.println("This number is too small.");
			}
		}
		System.out.println("Congratulations, that is the right number");
	}
}
```

```
Guess a number between 1 and 100
Your Guess:
50
This number is too large.
Your Guess:
12a
Exception in thread "main" java.util.InputMismatchException
	at java.util.Scanner.throwFor(Scanner.java:909)
	at java.util.Scanner.next(Scanner.java:1530)
	at java.util.Scanner.nextInt(Scanner.java:2160)
	at java.util.Scanner.nextInt(Scanner.java:2119)
	at at.java.GuessNumber.main(GuessNumber.java:13)
```



```
public class GuessNumber {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in);
                int guessedNumber = 0;
                int randomNumber = (int) ((Math.random() * 100) + 1);
                System.out.println("Guess a number between 1 and 100");
                while (guessedNumber != randomNumber) {
                        System.out.println("Your Guess:");
                        try {
                                guessedNumber = scanner.nextInt();
                                if (guessedNumber > randomNumber) {
                                        System.out.println("This number is too large.");
                                } else if (guessedNumber < randomNumber) {
                                        System.out.println("This number is too small.");
                                }
                        } catch (InputMismatchException ex) {
                                scanner.next(); //remove the invalid data from stream
                                System.out.println("Please stop talking nonsense.");
                        }

                }
                System.out.println("Congratulations, that is the right number");
        }
}
```

```
Guess a number between 1 and 100
Your Guess:
blablaIHateJava
Please stop talking nonsense.
Your Guess:
```



Unterschiedliche Exceptions

- Es können mehrere Exceptions von verschiedenen Typen in einem Try-Catch Block gefangen werden

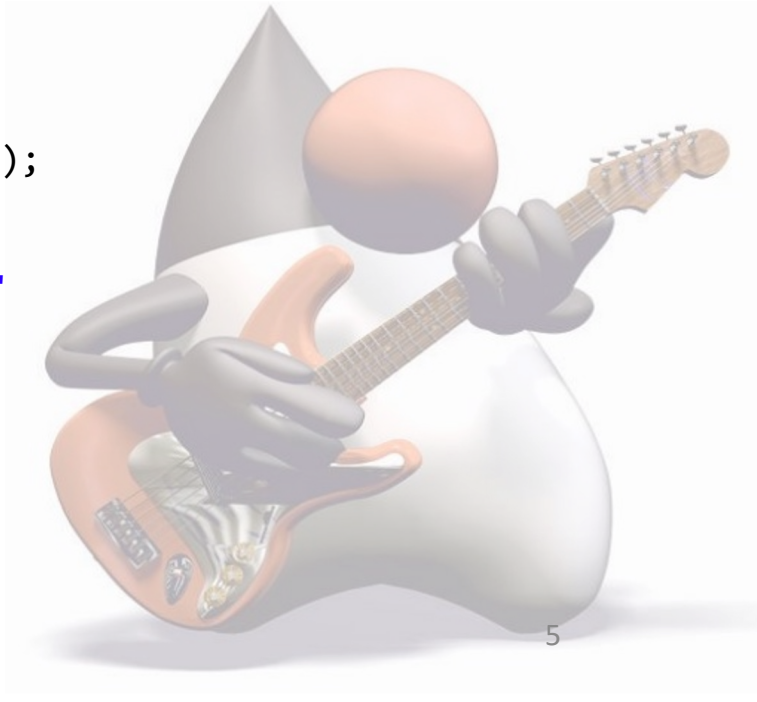
```
public static void main(String[] args) {
        try {
                // do code in which Exceptions can occur
        } catch (IOException ex) {
                // do error handling for IOExceptions
        } catch (IllegalArgumentException ex){
                // do error handling for IllegalArgumentExceptions
        }
}
```

Checked Exceptions

- Darum ***MUSS*** sich der Programmierer kümmern, sonst lässt sich das Programm nicht kompilieren

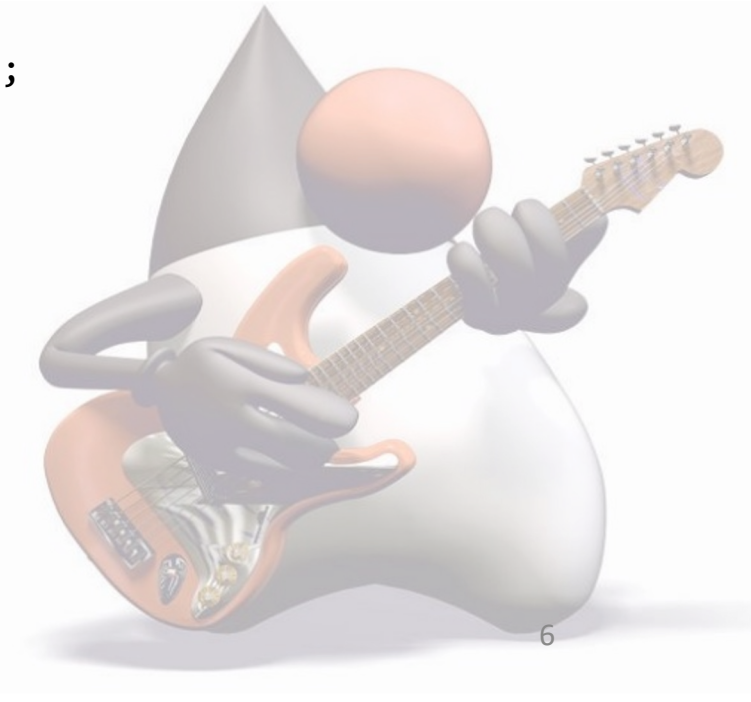
```
public class JustAClass {

        public static void main(String[] args) {
                JustAClass jac = new JustAClass();
                jac.parseDate("01.01.2011");
        }

        public Date parseDate(String dateString) {
                SimpleDateFormat format = new SimpleDateFormat("dd.MM.yyyy");
                Date result = null;
                        result = format.parse(dateString);
                        System.out.println("The date was not in format + "
                                                + format.toPattern());
                return result;
        }
}
```

Unhandled exception type ParseException



Exception Handling

```
public class JustAClass {

        public static void main(String[] args) {
                JustAClass jac = new JustAClass();
                jac.parseDate("01.01.2011");

        }

        public Date parseDate(String dateString) {
                SimpleDateFormat format = new SimpleDateFormat("dd.MM.yyyy");
                Date result = null;
                try {
                        result = format.parse(dateString);
                } catch (ParseException ex) {
                        System.out.println("The date was not in format + "
                                        + format.toPattern());
                }

                return result;
        }
}
```



Wir könnten die Methode auch in der aufrufenden Methode fangen. Dazu muss nur deklarieren dass parseDate() eine ParseException wirft.

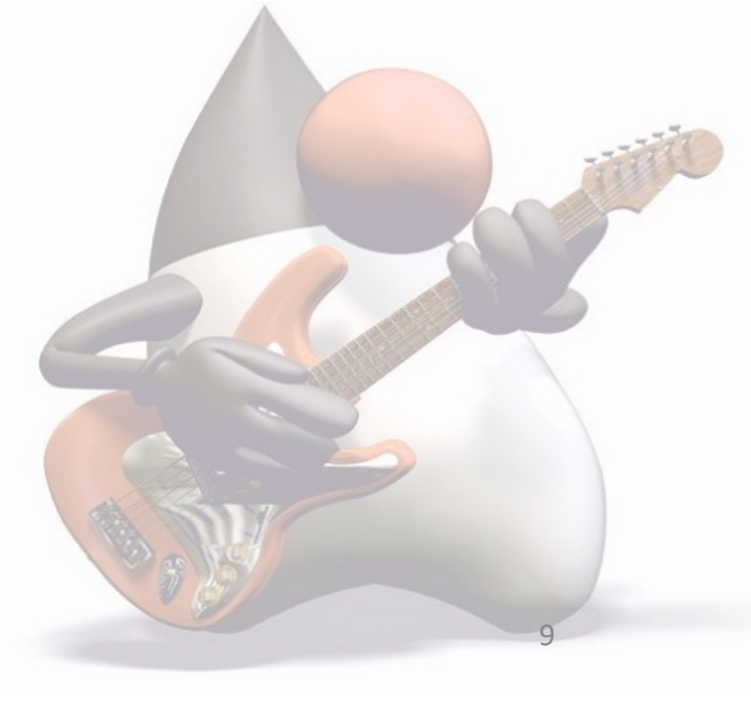
```
public class JustAClass {

        public static void main(String[] args) {
                JustAClass jac = new JustAClass();
                try {
                        jac.parseDate("01.01.2011");
                } catch (ParseException ex){
                        System.out
                        .println("The date was not in the right format");
                }

        }

        public Date parseDate(String dateString) throws ParseException {
                SimpleDateFormat format = new SimpleDateFormat("dd.MM.yyyy");
                Date result = null;
                result = format.parse(dateString);
                return result;
        }
}
```

Handle-Or-Declare Rule

- Wenn eine Methode aufgerufen wird die eine Exception wirft muss man eines von zwei Dingen tun.
  - mittels Try-Catch Block die Exception fangen und behandeln
  - throws in Methoden Signatur, um zu deklarieren, dass die Methode diese Exception wirft

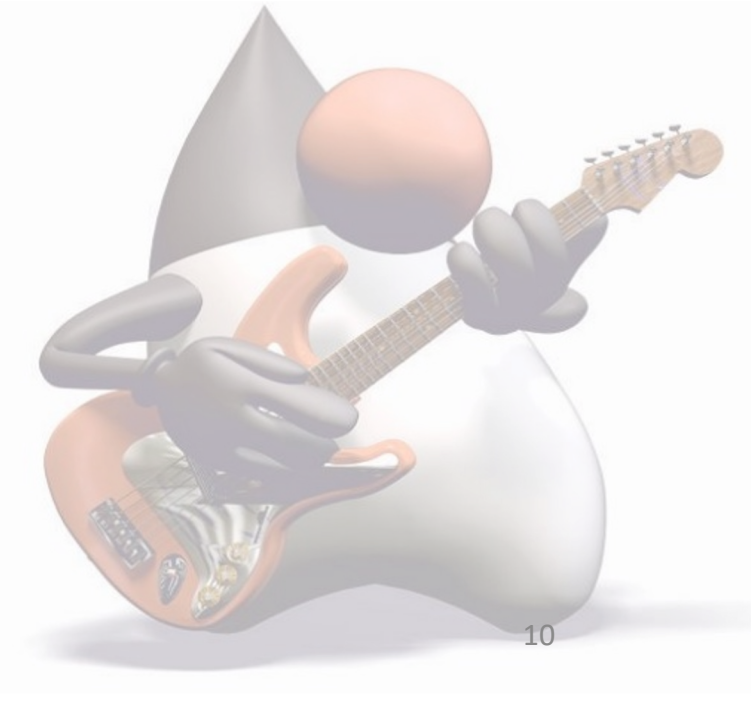
Finally Block wird immer ausgeführt

```
public class JustAClass {
        public static void main(String[] args)  {
                BufferedReader reader = null;
                try {
                        reader = new BufferedReader(new FileReader("C:\\Windows\\win.ini"));
                        String line = "";
                        while (null != (line = reader.readLine())) {
                                System.out.println(line);
                        }
                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                } finally {
                        if (reader!=null){
                                try {
                                        reader.close();
                                } catch (IOException e) {
                                        e.printStackTrace();
                                }
                        }
                }
        }
}
```



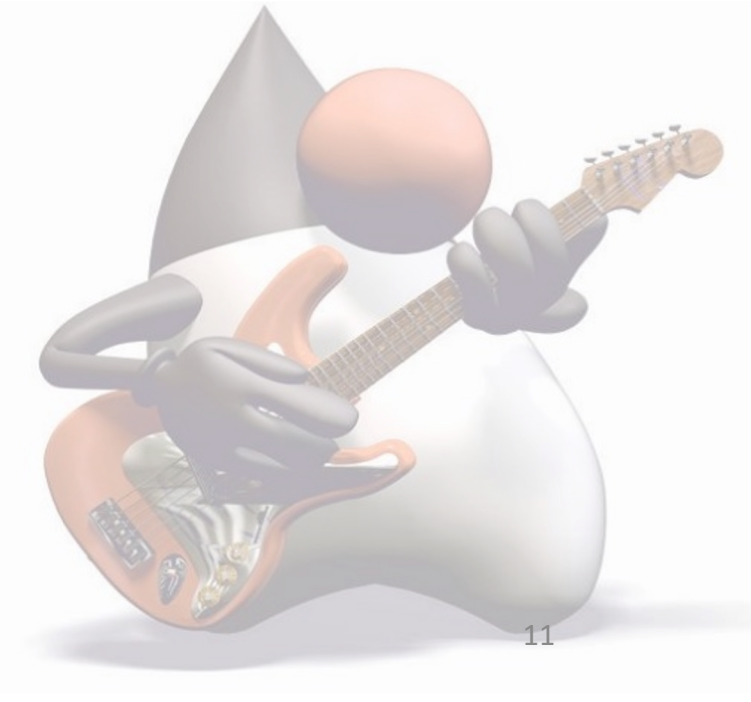
Eigene Exception Klassen erstellen

```
public class OverdraftException extends Exception{
        BigDecimal amount;

        public OverdraftException(String message, BigDecimal amount){
                super(message);
                this.amount = amount;
        }

        public BigDecimal getAmount() {
                return amount;
        }
}
```



Eigene Exception werfen

```
public boolean withdraw(double amount) throws OverdraftException {
        if (balance - amount < overdraftLimit) {
                throw new OverdraftException("Overdraft limit reached.",
                                        -overdraftLimit + balance - amount);
        }
        balance -= amount;
        return true;
}
```



Übungen

- Baue ein sinnvolles Exception Handling in Übung 6 ein.
- Implementiere eine UeberziehungException als Subclass von Exception
- Wirf die UeberziehungException wenn die Methode abheben nicht erfolgreich abläuft.