Exceptions

Voronezh, 2018

T · · Systems

Вопросы лекции

- Что такое исключение?
- Каких видов бывают исключения?
- Как обработать исключение?

Пример

```
1. public class Main {
2.     public static void main(String[] args) {
3.         for (int i = 0; i <= args.length; i++) {
4.             System.out.println(args[i]);
5.         }
6.     }
7. }
```

Пример

```
1. public class Main {
2.     public static void main(String[] args) {
3.         for (int i = 0; i <= args.length; i++) {
4.             System.out.println(args[i]);
5.         }
6.     }
7. }
```

```
>java Main hello world
```

T · · Systems

Пример

```
1. public class Main {
2.     public static void main(String[] args) {
3.         for (int i = 0; i <= args.length; i++) {
4.             System.out.println(args[i]);
5.         }
6.     }
7. }
```

```
>java Main hello world

hello
world
Exception in thread "main"
    java.lang.ArrayIndexOutOfBoundsException: 2
        at Main.main(Main.java:4)
```

T · · Systems

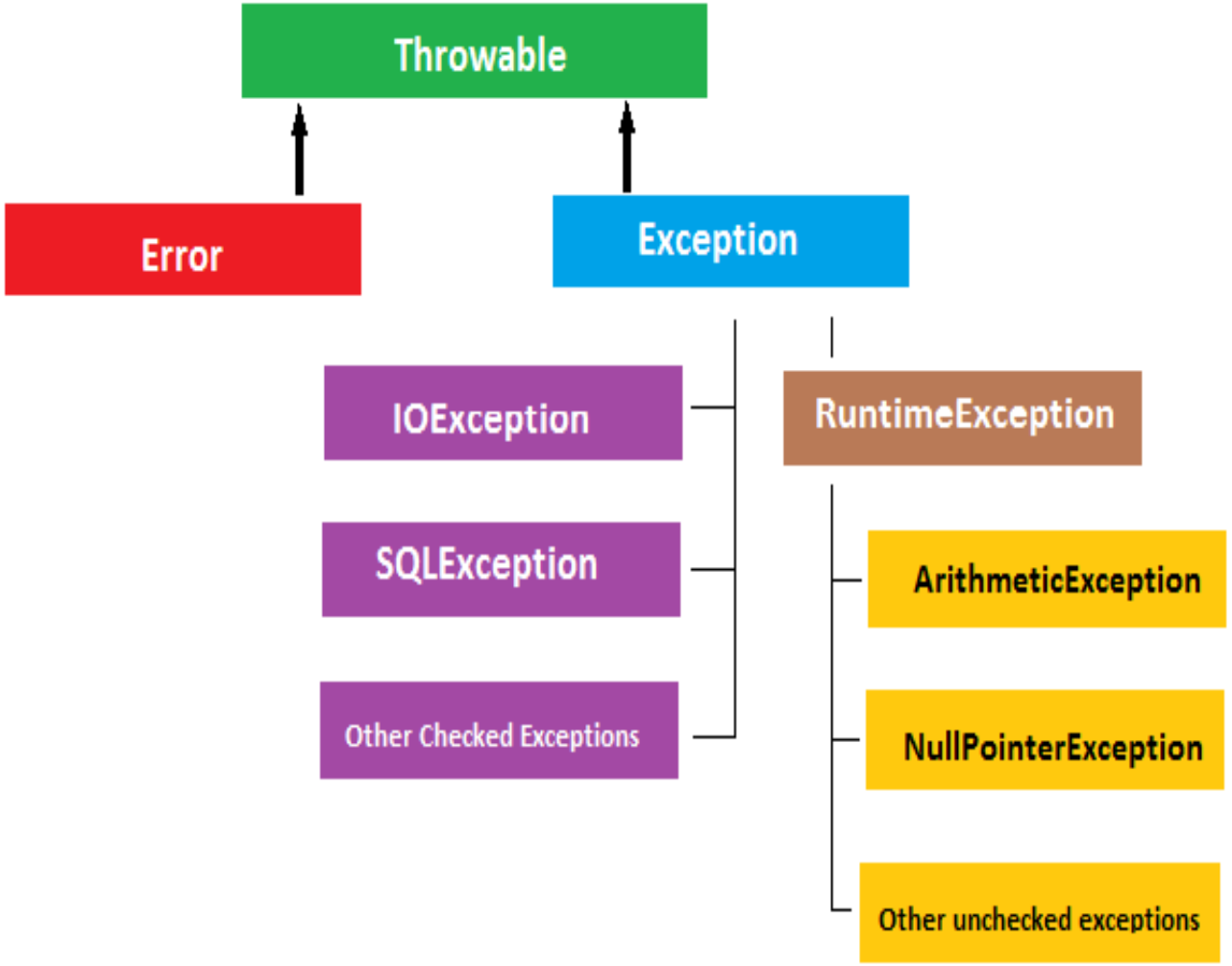
Что такое Exception?

- Исключение (Exception) – событие, происходящее в процессе выполнения программы, которое нарушает нормальную последовательность операций

- Применяются для:
  - Информирование о произошедшей ошибке
  - Запрос помощи в непредусмотренной ситуации

- Типы:
  - Checked exceptions
  - Unchecked / runtime exceptions
  - Errors

Иерархия исключений в Java

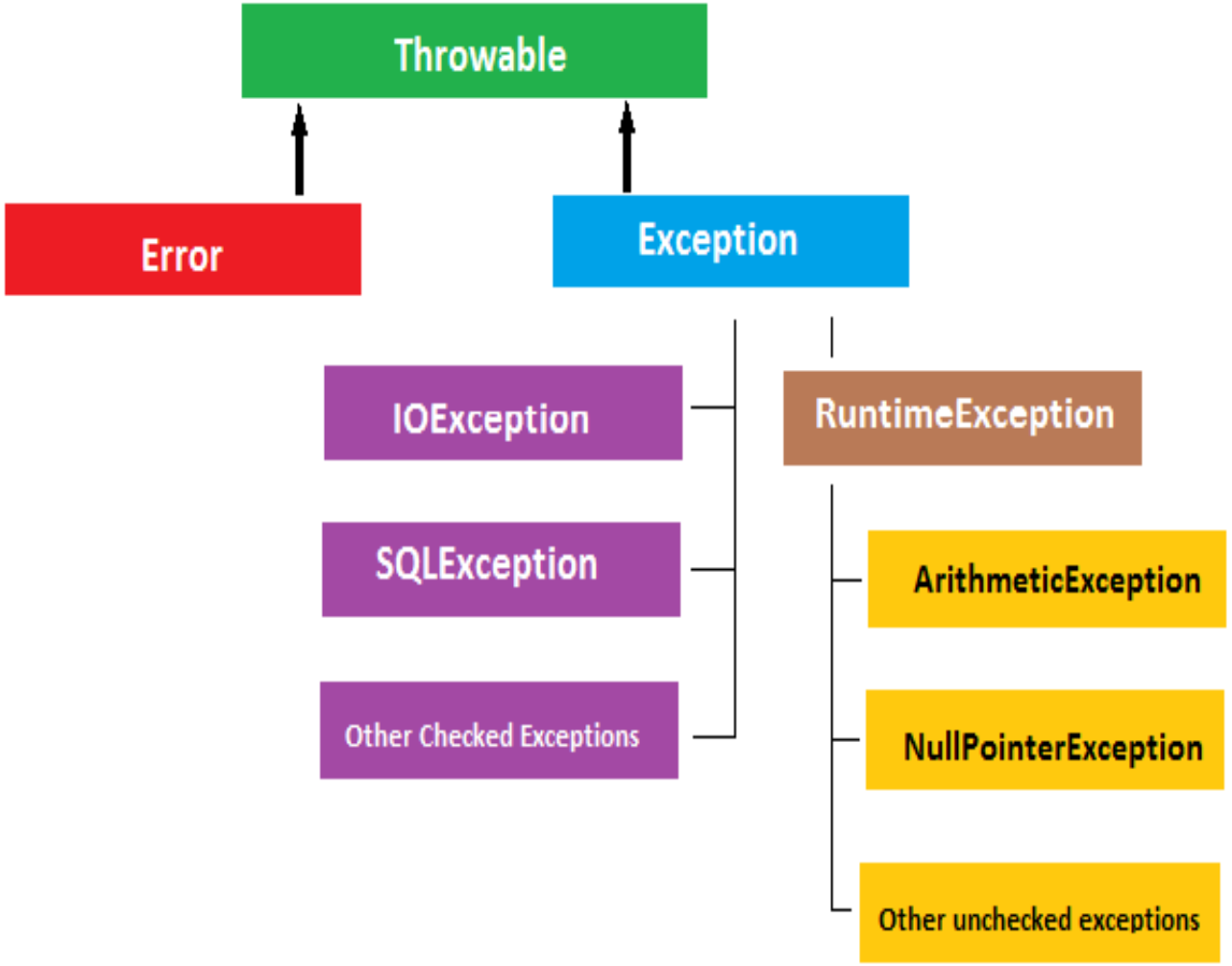


T · · Systems

Errors

- Потомки `java.lang.Error`
- Внешние по отношению к приложению исключительные ситуации, как правило, приложение не может предусмотреть или восстановиться после возникновения таких исключений.
- Например, приложение успешно открывает файл, но из-за неправильной работы ОС не может его прочитать, в этом случае сбрасывается `java.io.IOError`
- Обработка нужна только для информирования пользователя

Иерархия исключений в Java



Checked Exceptions

- Потомки `java.lang.Exception`

- Обязательно:
  - обрабатывать
  - объявлять в сигнатуре метода:

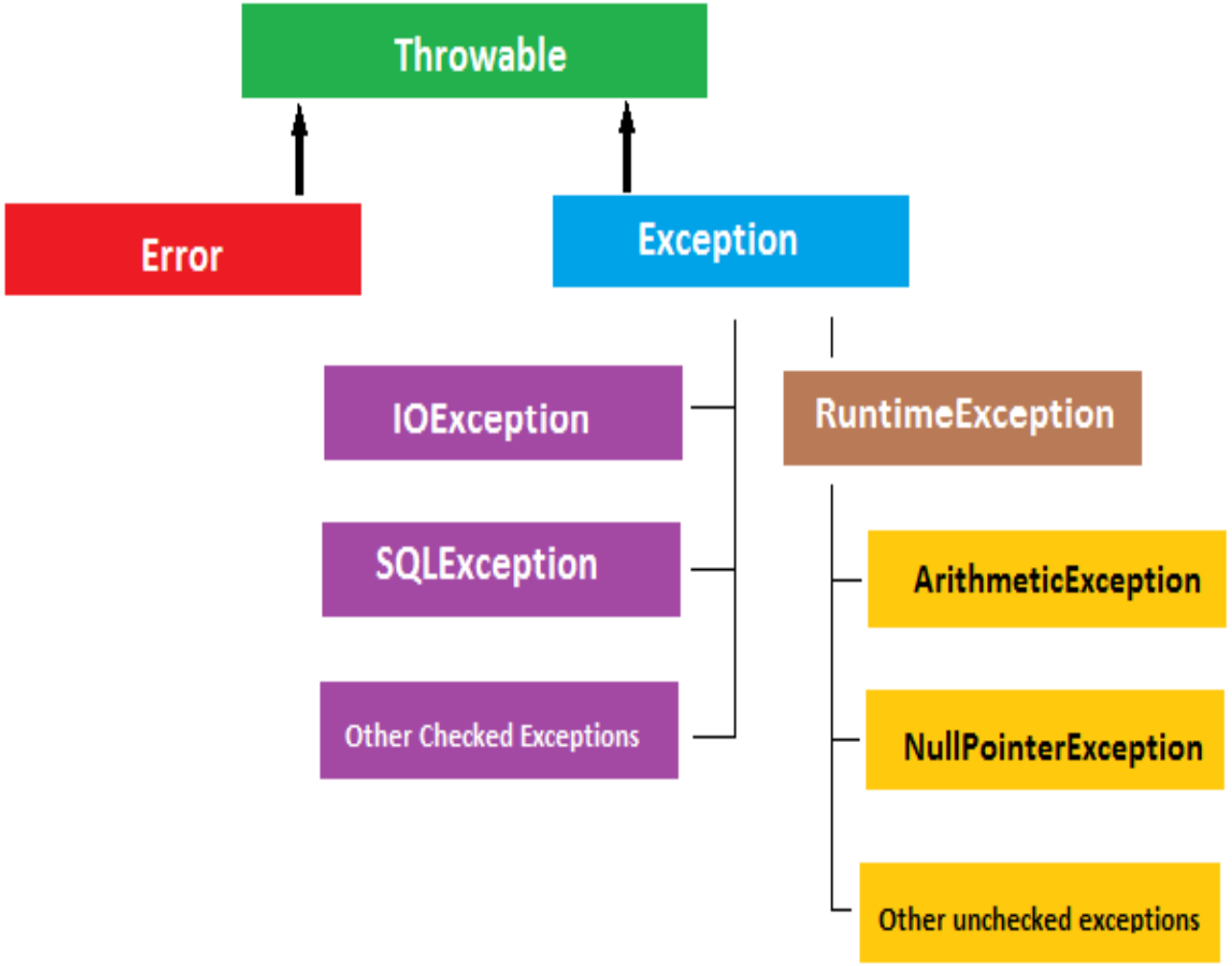
```
public void doSomething() throws SomeException
```

  - сбрасывать это исключение в теле метода:

```
if (isGoodValue) {
    throw new SomeException();
}
```

  - обрабатывать при вызове метода или перебрасывать в метод уровнем выше

Иерархия исключений в Java



T · · Systems

Runtime Exceptions

- Потомки `java.lang.RuntimeException`

- Необязательно:
  - обрабатывать
  - объявлять в сигнатуре метода
  - обрабатывать при вызове метода

- Сбросить runtime exception можно в любом месте:

```
throw new IllegalArgumentException("argument cannot be null")
```

- Обработка нужна, когда можно продолжить работу:

```
String text = "text";
try {
    System.out.println(text.substring(5));
} catch (StringIndexOutOfBoundsException e) {
    text = DEFAULT_VALUE;
}
```

T · · Systems

Создание и использование

- Типы наследуются от класса **`java.lang.Exception`** или **`java.lang.RuntimeException`**

```
public class MyException extends Exception {
  public MyException() {  super();  }
  public MyException(String message, Throwable cause) {
          super(message, cause);  }
  public MyException(String message) { super(message); }
  public MyException(Throwable cause) { super(cause); }
}

class A {
  void doItImmediately() throws MyException {
    // do smth
    if (isTrue) {
      throw new MyException("Cannot do it immediately! I want go
          away!");
} } }
```

T · · Systems

Обработка исключительных ситуаций (1)

- Для обработки используется блок **try-catch-finally**:

```
try {
    aMethodThrowingPossibleException()
} catch (PossibleException1 e1) {
    e.printStackTrace()
} catch (PossibleException2 e2) {
    e.printStackTrace()
} finally {
    // perform final processing
}
```

- Необработанное исключение поднимается на уровень выше (в вызвавший метод)

- Если исключение не будет обработано в методе **main()**, работа виртуальной машины будет завершена

Обработка исключительных ситуаций (2)

- Код в блоке `finally` будет выполнен в любом случае, даже если:
  - исключение произошло
  - исключение не произошло
  - внутри блока `catch` была использована команда `return`

- `try – catch – finally`:
  - Может использоваться без `finally`, когда нет обязательного к исполнению в любом случае кода
  - Может использоваться без `catch`, когда не предполагается обработка исключения, но есть обязательный к исполнению код
  - Может содержать несколько `catch`, расположенных иерархично от более узкого (потомка) к более широкому (родителю) исключению

Обработка исключительных ситуаций (3)

- До Java 7:

```
static String readFirstLineFromFile(String path) throws IOException {
    BufferedReader br = new BufferedReader(new FileReader(path));
    try {
        return br.readLine();
    } finally {
        if (br != null) br.close();
    }
}
```

- Начиная с Java 7 появился **try**-with-resources:

```
static String readFirstLineFromFile(String path) throws IOException {
 try (BufferedReader br = new BufferedReader(new FileReader(path))) {
        return br.readLine();
    }
}
```

T · · Systems

Пример обработки

```
1. public class Main {
2.     public static void main(String... args) {
3.         try {
4.             for (int i = 0; i <= args.length; i++) {
5.                 System.out.println(args[i]);
6.             }
7.         } catch (ArrayIndexOutOfBoundsException e) {
8.             System.err.println("Wrong index!");
9.         }
10.    }
11.}
```

```
>java Main hello world

hello
world
Wrong index!
```

Best Practice (1)

- Желательно использовать имеющиеся исключения, в редких случаях создавая собственные:

  `ConnectionFailedException` – существует `ConnectException`

  `OutOfMoneyException` – хорошее бизнес-исключение

- Исключения типа **`Error`** лучше не обрабатывать

- Использовать только там, где это необходимо

Best Practice

- Checked exceptions – для восстанавливаемых ситуаций, unchecked exceptions – для ошибок программы
- Избегать ненужного использования checked exceptions (для проверки состояний и т.д.)
- (*)Документировать все исключения, бросаемые в методах:
  - `@throws`
  - `throws XXXException`
- Добавлять информацию в сообщения

T · · Systems

try, catch, throw, throws, finally

1. Часть кода, которая может «бросить» исключение, заключается в **try** (как бы «попытаться выполнить это»)

2. Если возникает ошибка, система возбуждает (**throw**), или иначе говоря «бросает» исключение.

3. В зависимости от типа исключения, необходимо его «поймать» (**catch**) и обработать в соответсвующем блоке catch или передатъ обработчику по умолчанию **finally**.

Спасибо за внимание!

T · · Systems