# Исключения

- Что такое исключения?
- Какова их область применения?
- Какой синтаксис работы с ними?
- Какие существуют виды исключений и чем они отличаются?

# Основные моменты

# Определение

- Исключение - это ошибка (исключительная ситуация), возникающая во время выполнения программы.

**Ключевые слова:**

- **try** - ключевое слово, используемое для отметки начала блока кода, который потенциально может привести к ошибке;
- **catch** - ключевое слово для отметки начала блока кода, предназначенного для перехвата и обработки исключений;
- **finally** - ключевое слово для отметки начала блока кода, который исполняется после блока try (в случае отсутствия исключения) или catch (в случае возникновения исключения);
- **throw** - ключевое слово, которое служит для генерации исключений;
- **throws** - ключевое слово, которое прописывается в сигнатуре метода, и обозначает что метод потенциально может выбросить исключение указанного класс (либо его наследника).

# Синтаксис

## Пример конструкции:

```
try {
    // здесь описывается код, который потенциально может привести к генерации исключения
    someMethod();
} catch (SomeException e) { // в скобках указывается класс конкретного ожидаемого исключения
    // здесь описываются действия, направленные на обработку исключения
} finally {
    // здесь описывается код, который выполняется:
    // 1. после блока try в случае отсутствия исключений
    // 2. после блока catch в противном случае
}
```

# Синтаксис

**Обработка нескольких исключений:**

```
try {
    // здесь описывается код, который потенциально можети привести к генерации исключения
} catch (IOException e) {
    LOGGER.error(e.getMessage());
} catch (SQLException e) {
    LOGGER.error(e.getMessage());
}
```

**Альтернативный способ однотипной обработки:**

```
try {
    // здесь описывается код, который потенциально можети привести к генерации исключения
} catch (IOException | SQLException e) {
    LOGGER.error(e.getMessage());
}
```

# Синтаксис

**try with resources:**

```
try (AutoCloseable autoCloseable = new SomeAutoCloseable()) {
    // здесь описывается код, который потенциально можети привести к генерации исключения
} catch (Exception e) {
    LOGGER.error(e.getMessage());
}
```

**Реализация класса:**

```
class SomeAutoCloseable implements AutoCloseable {
    @Override
    public void close() throws Exception {
        // код для окончания работы с объектом (например, удаление файла)
    }
}
```

# Синтаксис

**Наследование методов бросающих исключения:**

```
public class SuperClass {
    public void start() throws IOException{
        throw new IOException("Not able to open file");
    }
}

class SubClass extends SuperClass{
    public void start() throws FileNotFoundException {
// FileNotFoundException - наследник IOException
        throw new FileNotFoundException("Not able to start");
    }
}
```

# Синтаксис

**Декларация исключения в сигнатуре метода:**

```
public void someMethodWhichThrowsException() throws Exception {
    throw new Exception();
}

public void someMethod() {
    try {
        someMethodWhichThrowsException();
    } catch (Exception e) {
        LOGGER.error(e.getMessage());
    }
}
```

# Синтаксис

- **checked** - исключения, которые должны обрабатываться блоком catch или описываться в сигнатуре метода;
- **unchecked** - исключения, которые не обязательно обрабатывать блоком catch или описывать в сигнатуре метода.

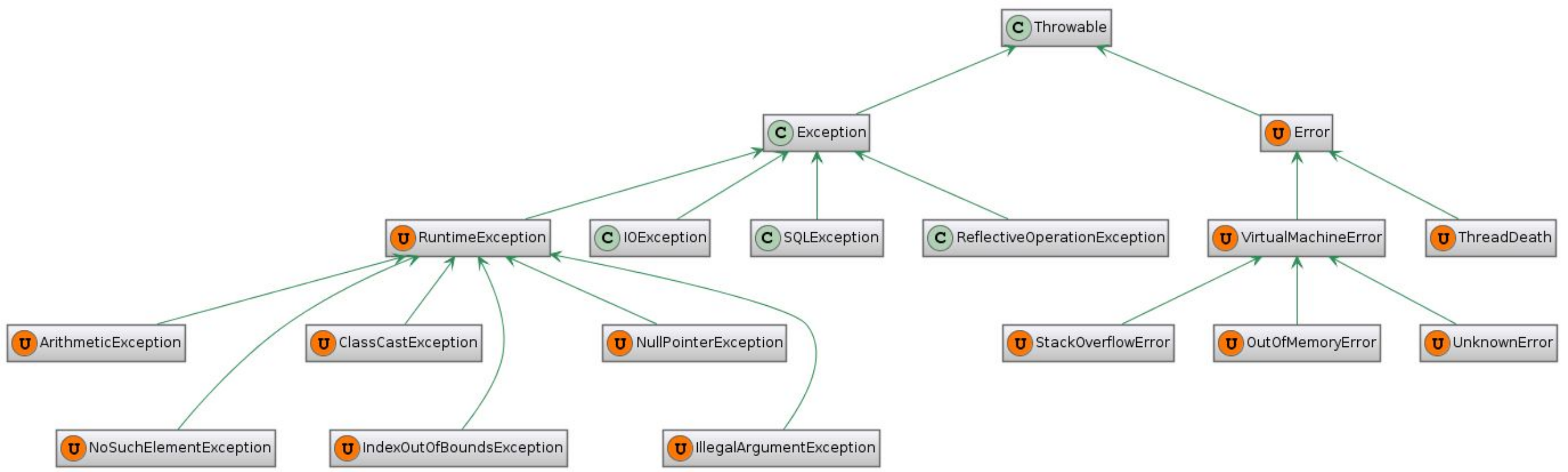
# Типы исключений

# Типы исключений

**Принцип выбора типа исключения:**

- **checked** - предсказуемая, но неизбежная ситуация, при возникновении которой имеется возможность разумно оправиться и логически целостно продолжить работу приложения;
- **unchecked** - все остальные ситуации.

**Иерархия наследования:**




# Типы исключений

**Создание собственного класса-исключения:**

```
public class SomeException extends Exception {
}
```

# Типы исключений

# Конец