# Exceptions

# Зачем нужны?

Исключения помогают писать программу с учетом непредвиденных ситуаций

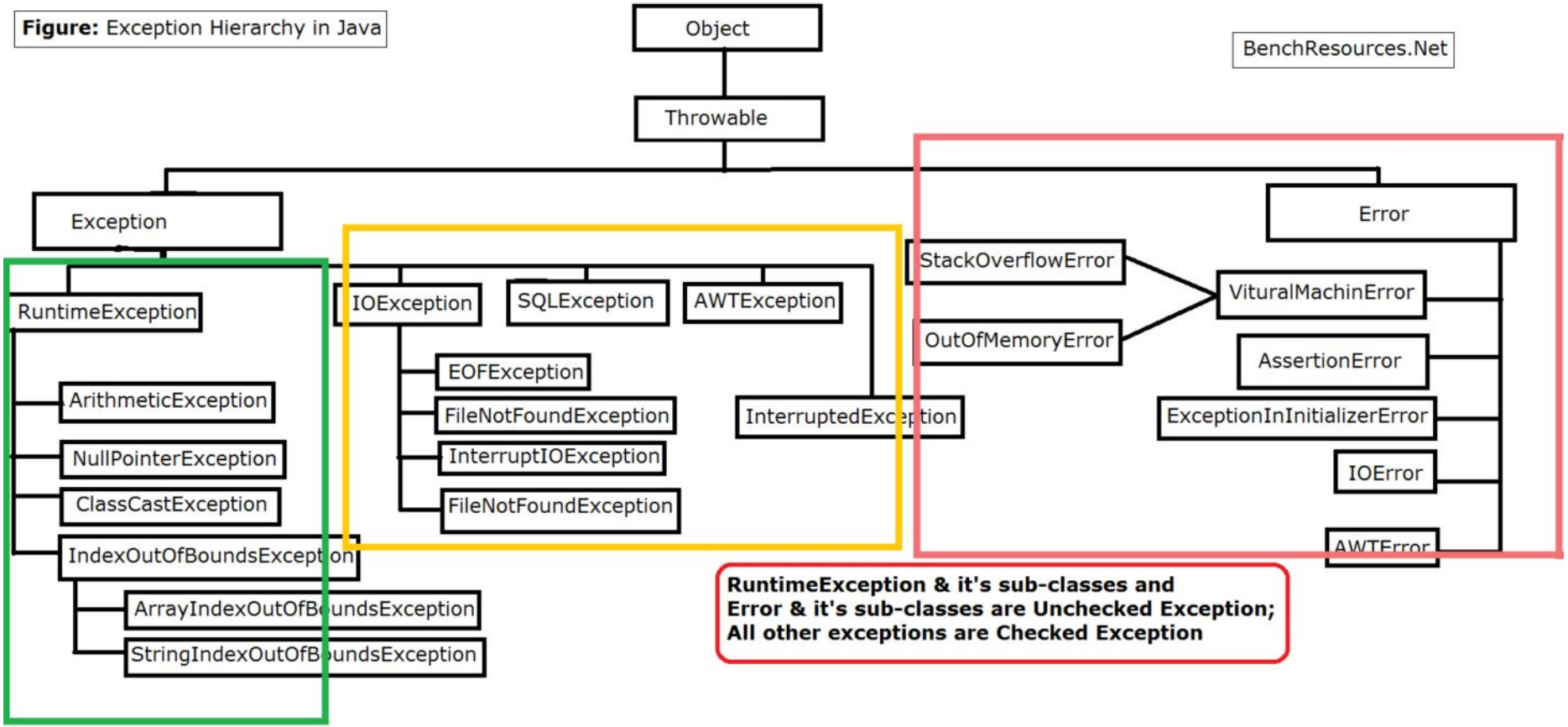
- Что делать, когда программа пытается открыть файл, которого нет?
- Что делать, когда идет запрос в базу, которая недоступна?
- Что делать, когда утром с мыслями о кружке кофе открыл шкаф, а кофе нет или сломалась кофемашина?

# Готовим кофе дома

# Исключительная ситуация

# Что такое Exceptions

## Exception - это класс



# Типы ошибок

**Непроверяемые** (Unchecked) - Error и RuntimeException

- Ошибки JVM
- Ошибки разработчика (деление на 0, выход за рамки массива)

**Проверяемые** (Checked) - **Всё** что не Error и не RuntimeException

- Ошибки доступа к ресурсам
- Любые другие созданные разработчиком, которые надо обработать

# Как создать своё проверяемое исключение?

```
public class CoffeeNotFoundException extends Exception {
    public CoffeeNotFoundException(String message) {
        super(message);
    }
}
```

вызывается конструктор Exception

- Наследуется от Exception
- Чтобы передать сообщение надо вызвать конструктор суперкласса

# Как бросаются исключения?

```
public String getCoffee() throws CoffeeNotFoundException
{
    System.out.println("Достать кофе из шкафа");
    if(coffeeBeans == null){
        throw new CoffeeNotFoundException("Контейнер с кофе пустой");
    }
    return coffeeBeans;
}
```

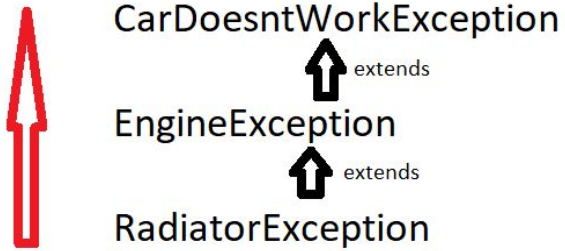
**throw** - оператор “бросания ошибки” в теле метода

**throws** - оператор, показывающий что метод пробрасывает ошибку тому, кто его использует (пользователь)

# Как обрабатывать ошибки?

```
try {
    // блок кода, в котором может возникнуть ошибка
} catch(MyException e) {
    // Что-то делаем когда возникает эксепшен типа MyException
} finally {
    // Что-то делаем всегда (почти всегда)
}
```

- блоков **catch** может быть несколько, но обработается только один
- **catch** обрабатывает ошибки по иерархии снизу вверх




- **finally** - необязательный блок
- **finally** может быть без catch
- **finally** не выполнится если будет System.exit(0)

# try with resources

```
try (/*создание ресурса*/) {
        // Работа с ресурсом
    } catch (FileNotFoundException e) {
        // работа с ошибками при работе с ресурсом
    }
}
```

- Не требует закрытие ресурса в finally
- Ресурс должен реализовывать интерфейс AutoCloseable

# Переопределение методов с исключениями

- При переопределении можно не указывать исключение, которое бросает супер-метод, если в переопределенном его нет
- Нельзя указывать эксепшен из другой иерархии или исключение из иерархии выше