Программирование на Java

5. Командная строка

Глухих Михаил Игоревич
mailto: glukhikh@mail.ru

Пример: перекодировщик

- Приложение должно запускаться так:

```
java –jar appname.jar –ie InputEncoding
-oe OutputEncoding InputName OutputName
```

Пример: перекодировщик

- Приложение должно запускаться так:
  ```
  java –jar appname.jar –ie InputEncoding
  -oe OutputEncoding InputName OutputName
  ```

- Здесь
  - InputEncoding – кодировка входного файла
  - OutputEncoding – требуемая кодировка
  - InputName – имя входного файла
  - OutputName – требуемое имя выходного файла

Пример: перекодировщик

- Приложение должно запускаться так:
  ```
  java –jar appname.jar –ie InputEncoding
  -oe OutputEncoding InputName OutputName
  ```

- Аргументы командной строки
  - –ie (ключ опции)
  - InputEncoding
  - –oe (ключ опции)
  - OutputEncoding
  - InputName
  - OutputName

Разбор командной строки

- Вариант «В лоб»

```
public static void main(String[] args) {
    ...
}
```

- И далее читаем массив args и анализируем его

Разбор командной строки

- Использование готовой библиотеки
  - Пример: org.kohsuke.args4j
- Возможности
  - Поддержка «опций» командной строки
  - Поддержка «необходимых» и «опциональных» параметров
  - Вывод инструкции (Usage)
  - …

Пример разбора командной строки

- См. part2.recode.java.RecoderLauncher

Системы сборки проекта

- Maven
  - pom.xml
- Gradle

Системы сборки проекта

- Преимущества
  - Не зависят от IDE
  - Следят за зависимостями между библиотеками
  - Не требуют выкладывания библиотек в репозиторий

Перекодировка: как?

- Необходим поток, поддерживающий кодировку
  - InputStream
    - InputStreamReader
      - BufferedReader

Перекодировка: как?

- Необходим поток, поддерживающий кодировку
  - InputStream = bytes
    - **InputStreamReader = characters**
      - BufferedReader = strings

Перекодировка: реализация

```
public int recode(InputStream in, OutputStream out)
           throws IOException {
    try (InputStreamReader reader =
          new InputStreamReader(in, charsetInput)) {
        try (OutputStreamWriter writer =
             new OutputStreamWriter(out, charsetOutput)) {
            int sym = reader.read();
            int count = 0;
            while (sym != -1) {
                writer.write(sym);
                count++;
                sym = reader.read();
            }
            return count;
        }
    }
}
```

Перекодировка: Котлин

```
fun recode(`in`: InputStream, out: OutputStream): Int {
    InputStreamReader(`in`, charsetInput).use { reader ->
        OutputStreamWriter(out, charsetOutput).use {
            writer ->
            var sym = reader.read()
            var count = 0
            while (sym != -1) {
                writer.write(sym)
                count++
                sym = reader.read()
            }
            return count
        }
    }
}
```

Класс Throwable

- Объект "исключение" содержит информацию о произошедшей ошибке
- При возникновении исключения мы поднимаемся наверх по стеку вызовов функций, пока не найдем блок обработки исключения
- Исключения используются в тех случаях, когда функция не может адекватно обработать возникшую ситуацию

Работа с исключениями в Java

- Конструкция формирования

```
throw new IllegalArgumentException(
    "Аргумент неправильный!");
```

- Блок обработки исключения

```
try {
} catch (IllegalArgumentException ex) {
} catch (UnsupportedOperationException ex) {
} catch (RuntimeException ex) {
} catch (Exception ex) {
} finally {
}
```

- Признак возможности возникновения исключения

```
public void f() throws SomeException { ... }
```

Совместная обработка исключений в JDK7

```
try {
} catch (IllegalArgumentException|
         UnsupportedOperationException ex) {
} finally {
}
```

Блок finally

- Всегда выполняется после окончания блока try/catch:
  - в случае успешного завершения блока try
  - в случае успешной обработки одного из исключений в блоке catch
  - в случае возникновения необрабатываемого исключения
- Используется обычно для выполнения определенных завершающих действий (например, закрытия файла)

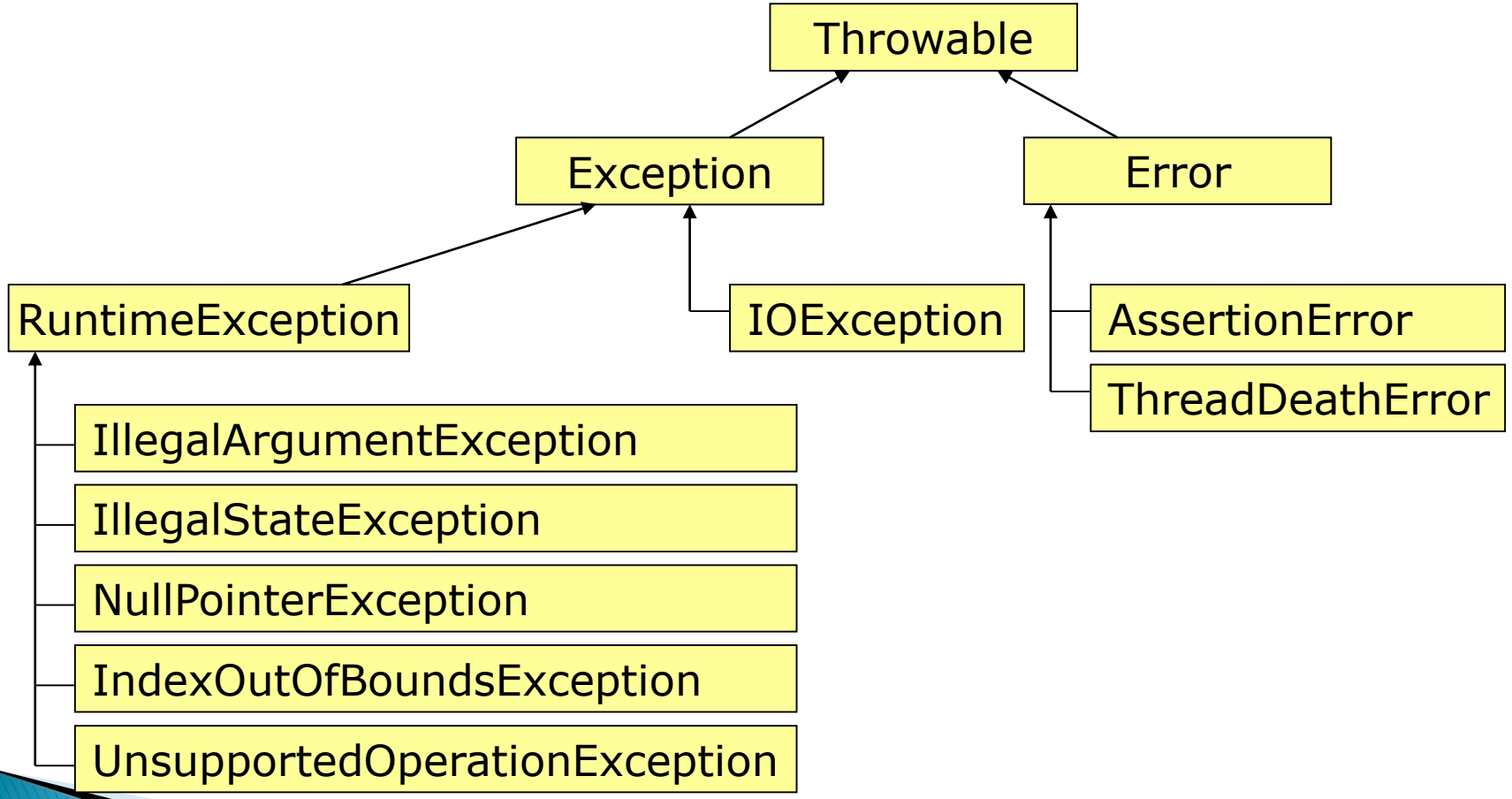
Закрытие ресурсов в JDK 6

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

Закрытие ресурсов в JDK 7

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

Иерархия исключений в Java



Иерархия исключений в Java

- Throwable – любое исключение
  - Error – исключения, которые обычно не обрабатывают
    - AssertionError – не сработала программная проверка
  - Exception – контролируемые исключения (нужно ловить или указывать throws)
    - RuntimeException – неконтролируемые исключения (можно не указывать throws)

По смыслу

- Контролируемое (checked) исключение – то, что может случиться при нормальной работе программы

- Неконтролируемое (unchecked) – то, что не должно случаться при нормальной работе (программная ошибка)

Основные методы исключений

- Конструкторы

```
new Exception("Описание")
new Exception(throwable)
new Exception("Описание", throwable)
```

- assert условие : "Описание" – программная проверка
- getMessage() – получить сообщение
- printStackTrace() – распечатать стек вызовов

Исключение в лямбде

```
public void write(String[] lines)
                        throws IOException {
    final BufferedWriter writer =
        new BufferedWriter(
            new FileWriter(file));
    Arrays.stream(lines).forEach(
        writer::write);
    writer.close();
}
```

Запуск приложения

```
Recoder recoder = new Recoder(inputEncoding,
                              outputEncoding);
try {
    int result = recoder.recode(inputFileName,
                                outputFileName);
    System.out.println(
        "Total of " + result + " symbols recoded");
} catch (IOException e) {
    System.err.println(e.getMessage());
}
```

Тестирование приложения

- См. test/part2.recode.RecoderTest

Итоги

- Рассмотрено
  - Командная строка
    - Библиотеки разбора
  - Обработка исключений
  - Потоки и кодировки
  - Тестирование приложений
- Далее
  - Система классов Java