Exceptions and Errors

```
git remote add origin git@github.com:javaAutoTester/QuestionList.git
git push -u origin master
```

Что для программы является исключительной ситуацией? Какие способы обработки ошибок в программах вы знаете?

Исключительная ситуация (ИС)
1. Выполнение действий, которые невозможны с точки зрения используемых аксиом.
Н/п: деление на ноль.
2. Выполнение действий, которые невозможны в данной ситуации.
Н/п: получение доступа к несуществующему элементу массива; попытка записи в файл, который используется другим пользователем; продолжение работы программы при исчерпании оперативной памяти.
Выполнение дальнейших действий при возникновении некоторых ИС может привести к потере или искажению данных.
Действия программы в ИС:
1. Прекращение работы программы.
2. Обработка ситуации согласно алгоритма, предложенного разработчиком.

Exceptions and Errors

Что такое исключение для Java-программы? Что значит “программа выбросила исключение”?

Исключения для Java-программы – это объекты класса Throwable, которые создаются в момент возникновения исключительной ситуации (ИС). Все ИС в Java структурированы, поэтому в зависимости от типа возникшей ИС создается соответствующий наследник класса Throwable. По имени класса наследника, можно определить к какому типу относится возникшая ИС.
Для оповещения пользователей о возникшей ИС используется механизм “выбрасывания исключений”.

“Выбросить исключение”- значит вывести оповещение, в котором указан тип возникшей ИС и номер строки кода, которая привела к возникновению ИС.

Приведите иерархию классов-исключений, делящую исключения на проверяемые и непроверяемые. В чем особенности проверяемых и непроверяемых исключений?

Рассмотрим упрощенную иерархию исключений в Java.

Throwable

Exception

Исключительные ситуации, с которыми можно жить.
Данные ИС можно обрабатывать, путем указания программе конкретных действий в данной ситуации

Checked
(IOException, SQLException, ...)

Данные исключения проверяются на этапе компилляции. Компиллятор подсказывает, что выполнение данного действия может привести к возникновению ИС , т.е. разработчики применяемых классов уже предусмотрели возможность появления такой ситуации. Программист в этом случае должен предпринять определенные шаги: добавить **throws** в сигнатуру метода, или обработать исключение в блоке **try...catch.**

Uncheked
(RuntimeException)

Исключения не проверяются компиллятором(они не могут быть проверены на данном этапе). Эти исключения выбрасываться в процессе нормальной работы JVM. Программист должен сам предугодать возникновение данных ИС (н/п: деление на ноль, обращение к несуществующему элементу Массива) и либо недопустить их возникновение, либо предусмотреть их обработку **try...catch** (крайний вариант: **throws**)

Error

ИС, при возникновении которых дальнейшая корректная работа программы невозможна. Требуется пересмотр кода и недопущение данных ИС

Uncheked

Данные критические ИС невозможно предвидеть на этапе компилляции

Exceptions and Errors

Опишите ситуации, когда исключения выбрасываются виртуальной машиной(автоматически), и когда необходимо их выбрасывать вручную?

Учитывая схему выше, все исключения, которые описаны и структурированы в Java выбрасываются автоматически. В случае проверяемых исключений, компиллятор потребует явно указать, как действовать программе при возникновении ИС. В случае непроверяемых исключений и ошибок, данного указания не требуется. Ситуации, в которых выбрасываются данные исключения прописаны в логике Java и служат для применения самой Java. Программисту предоставляется лишь некоторая возможность управления.

Но также сущенствует возможность создания своих истключений. Например, мы можем создать свое исключение, которое будет выбрасываться в случае ввода пользователем некорректных данных. В этом случае исключение выбрасывается вручную.

Exceptions and Errors

Q1. Объясните работу оператора try-catch-finally. Когда данный оператор следует использовать?
Q2. Сколько блоков catch может соответствовать одному блоку try? Можно ли вкладывать блоки try друг в друга, можно ли вложить блок try в catch или finally?

Q1. Добавление в сигнатуру метода ключевого слова **throws** указывает на то, что возможны ситуации, когда данный метод выбросит указанное исключение. Причем при возникновении такой ситуации программа выбросит исключение и прекратит работу.
Оператор **try-catch-finally** позволяет изменить ход программы при возникновении ИС не прерывая ее исполнение. Правилом является использование оператора try-catch-finally для оборабоки исключений, а не использование throws.
Q2. Одному блоку try может соответствовать неограниченное количество блоков catch.
Блоки try можно вкладывать друг в друга, в блоки catch и в блоки finally. Если во внутреннем блоке catch нет исключения, выброшенного блоком try, то Java ищет выброшенное исключение в наружных блоках catch. Если не находит там, то завершает работу программы. Все блоки final при этом выполняются.

Как происходит обработка исключений, выброшенных внутренним блоком try, если среди его блоков catch нет подходящего?

```
public static void main(String[] args) throws IOException {
System.out.println("1");
String file = null;
File f = null;
try {
        try { f = new File(file);
            } catch (ArithmeticException e) {
                                    System.out.println("catch: Parametr \"file\" is null");
             }finally { file = "example.txt";
                         System.out.println("finally: File example.txt created");
                        }
  System.out.println("2");
  f = new File(file);
  System.out.println("3");
  f.createNewFile();
  System.out.println("4");
}catch(IOException e) { System.out.println("catch: Trap for NullPointerException");
                        e.printStackTrace();
                        String s = null;
                        try { System.out.println("6"+s.length());
                        }catch(NullPointerException ex) {
                               ex.printStackTrace();
                        }
}finally { System.out.println("finally: 555");}
System.out.println("5");
}
run:
1
finally: File example.txt created
finally: 555
Exception in thread "main" java.lang.NullPointerException
         at java.io.File.<init>(File.java:277)
         at by.htp.exeption.run.MainAppSecond.main(MainAppSecond.java:16)
```

В данном примере в первом внутреннем блоке try выбрасывается NullPointerException, т.к. file = null. Но такого исключения нет ни в одном блоке catch. Поэтому программа прекращает работу, выполнив операции во всех блоках finally. Последний оператор `System.out.println("5");` не выполнился.

Как происходит обработка исключений, выброшенных внутренним блоком try, если среди его блоков catch нет подходящего?

```
public static void main(String[] args) throws IOException {
System.out.println("1");
String file = null;
File f = null;
try {
        try { f = new File(file);
            } catch (ArithmeticException e) {
                                    System.out.println("catch: Parametr \"file\" is null");
             }finally { file = "example.txt";
                         System.out.println("finally: File example.txt created");
                        }
  System.out.println("2"); f = new File(file);
  System.out.println("3");f.createNewFile();System.out.println("4");
}catch(IOException | NullPointerException e) {
                         System.out.println("catch: Trap for NullPointerException");
                        e.printStackTrace();
                        String s = null;
                        try { System.out.println("6"+s.length());
                        }catch(NullPointerException ex) {
                                ex.printStackTrace();
                        }
}finally { System.out.println("finally: 555");} System.out.println("5");
}
run:
1
finally: File example.txt created
catch: Trap for NullPointerException
java.lang.NullPointerException
         at java.io.File.<init>(File.java:277)
         at by.htp.exeption.run.MainAppSecond.main(MainAppSecond.java:16)
java.lang.NullPointerException
         at by.htp.exeption.run.MainAppSecond.main(MainAppSecond.java:33)
finally: 555
5
```

См. предыдущий пример. Добавим NullPointerException во внешний catch блок. Т.к. все исключения были словлены, то программа выполнилась до конца, несмотря на два выброшенных исключения.

Что называют стеком операторов try? Как работает блок try с ресурсами? https://www.youtube.com/watch?v=CMy8I8VtPiw
В механизме try with resorses используемый русурс должен иметь метод close(), т.е. должен реализовывать интерфейс Closeable.