4. Семинар: обработка исключений

4.1. Инструментарий

— Презентация для преподавателя, ведущего семинар;
— Фон GeekBrains для проведения семинара в Zoom;
— JDK любая 11 версии и выше;
— IntelliJ IDEA Community Edition для практики и примеров используется IDEA.

4.2. Цели семинара

— Закрепить полученные на лекции знания об исключениях;
— Попрактиковаться в написании собственных классов исключений;
— Создание, выбрасывание и обработка исключений;
— Обработка исключений в стиле «до захвата ресурса»;
— Проброс исключений выше по стеку.

4.3. План-содержание

Что происходит	Время	Слайды	Описание
Организационный момент	5	1-5	Преподаватель ожидает студентов, поддерживает активность и коммуникацию в чате, озвучиает цели и планы на семинар. Важно упомянуть, что выполнение домашних заданий с лекции является, фактически, подготовкой к семинару
Quiz	5	6-18	Преподаватель задаёт вопросы викторины, через 30 секунд демонстрирует слайд-подсказку и ожидает ответов (6 вопросов, по минуте на ответ)
Рассмотрение ДЗ лекции	10	19-23	Преподаватель демонстрирует свой вариант решения домашнего задания с лекции, возможно, по предварительному опросу, демонстрирует и разбирает вариант решения одного из студентов
Вопросы и ответы	10	24	Преподаватель ожидает вопросов по теме прошедшей лекции, викторины и продемонстрированной работы
Задание 1	30	25-35	Сквозное задание, состоящее из объяснений в 6 пунктах, обязательных к исполнению. Всё задание выдаётся сразу целиком и является неделимым.

Что происходит	Время	Слайды	Описание
Перерыв (если нужен)	5	36	Преподаватель предлагает студентам перерыв на 5 минут (студенты голосуют)
Задание 2	40	37-49	Сквозное задание, состоящее из объяснений в 6 пунктах, обязательных к исполнению. Всё задание выдаётся сразу целиком и является неделимым.
Домашнее задание	5	50-51	Объясните домашнее задание, подведите итоги урока
Рефлексия	10	52-53	Преподаватель запрашивает обратную связь
Длительность	120		

4.4. Подробности

Организационный момент

- **Цель этапа:** Позитивно начать урок, создать комфортную среду для обучения.
- **Тайминг:** 3-5 минут.
- **Действия преподавателя:**
  - Запрашивает активность от аудитории в чате;
  - Презентует цели курса и семинара;
  - Презентует краткий план семинара и что студент научится делать.

Quiz

- **Цель этапа:** Вовлечение аудитории в обратную связь.
- **Тайминг:** 5-7 минут (4 вопроса, по минуте на ответ).
- **Действия преподавателя:**
  - Преподаватель задаёт вопросы викторины, представленные на слайдах презентации;
  - через 30 секунд демонстрирует слайд-подсказку и ожидает ответов.
- **Вопросы и ответы:**
  1. Перечисление – это: 2
     (a) массив
     (b) класс
     (c) объект
  2. Инкапсуляция с использованием внутренних классов: 2
     (a) остаётся неизменной
     (b) увеличивается
     (c) уменьшается
  3. Обработчик исключений – это объект, работающий 1
     (a) в специальном потоке
     (b) в специальном ресурсе
     (c) в специальный момент

4. Стектрейс - это 2
   (a) часть потока выполнения программы;
   (b) часть объекта исключения;
   (c) часть информации в окне отладчика.

Рассмотрение ДЗ

- **Цель этапа:** Пояснить не очевидные моменты в формулировке ДЗ с лекции, синхронизировать прочитанный на лекции материал к началу семинара.
- **Тайминг:** 15-20 минут.
- **Действия преподавателя:**
  - Преподаватель демонстрирует свой вариант решения домашнего задания из лекции;
  - возможно, по предварительному опросу, демонстрирует и разбирает вариант решения одного из студентов.
- **Домашнее задание из лекции:**
  - Напишите два наследника класса `Exception`: ошибка преобразования строки и ошибка преобразования столбца.
    Вариант решения

Листинг 1: Простые наследники

```
private static final class ColumnMismatchException extends RuntimeException {
   ColumnMismatchException(String message) {
      super("Columns exception: " + message);
   }
}

private static final class NumberIsNotNumberException extends RuntimeException {
   NumberIsNotNumberException(String message) {
      super("Not a number found: " + message);
   }
}
```

  - Разработайте исключения-наследники так, чтобы они информировали пользователя в формате ожидание/реальность.

Листинг 2: Составное сообщение

```
private static final class RowMismatchException extends RuntimeException {
   RowMismatchException(int expected, int current, String value) {
      super(String.format("Rows exception: expected %d rows. Received %d rows in '%s' string",
            expected, current, value));
   }
}
```

  - для проверки напишите программу, преобразующую квадратный массив целых чисел 5x5 в сумму чисел в этом массиве, при этом, программа должна выбросить исключение, если строк или столбцов в исходном массиве окажется не 5.
    Вариант решения представлен в приложении А

Вопросы и ответы

- **Ценность этапа** Вовлечение аудитории в обратную связь, пояснение неочевидных моментов в материале лекции и другой проделанной работе.
- **Тайминг** 5-15 минут
- **Действия преподавателя**
  - Преподаватель ожидает вопросов по теме прошедшей лекции, викторины и продемонстрированной работы;
  - Если преподаватель затрудняется с ответом, необходимо мягко предложить студенту ответить на его вопрос на следующем семинаре (и не забыть найти ответ на вопрос студента!);
  - Предложить и показать пути самостоятельного поиска студентом ответа на заданный вопрос;
  - Посоветовать литературу на тему заданного вопроса;
  - Дополнительно указать на то, что все сведения для выполнения домашнего задания, прохождения викторины и работы на семинаре были рассмотрены в методическом материале к этому или предыдущим урокам.

Задание 1

- **Ценность этапа** Написание почти полноценной механики по краткому ТЗ.
- **Тайминг** 25-30 минут.
- **Действия преподавателя**
  - Выдать задание студентам;
  - Подробно объяснить, что именно требуется от студентов, избегая упоминания конкретных языковых конструкций.
- **Задание**: Класс «Проверка логина и пароля».
  1. Создать статический метод который принимает на вход три параметра: `login`, `password` и `confirmPassword`.
  2. Длина `login` должна быть **меньше** 20 символов. Если `login` не соответствует этому требованию, необходимо выбросить `WrongLoginException`.
  3. Длина `password` должна быть **не меньше** 20 символов. Также `password` и `confirmPas` должны быть равны. Если `password` не соответствует этим требованиям, необходимо выбросить `WrongPasswordException`.
  4. `WrongPasswordException` и `WrongLoginException` – пользовательские классы исключения с двумя конструкторами –- один по умолчанию, второй принимает параметры исключения (неверные данные) и возвращает пользователю в виде «ожидалось/фактически».
  5. В основном классе программы необходимо по-разному обработать исключения.
  6. Метод возвращает `true`, если значения верны или `false` в противном случае.

  Вариант исполнения класса в приложении Б

  Вариант маршрута решения задачи

Листинг 3: Описание класса исключения логина

```
1 public static class WrongLoginException extends RuntimeException {
2    private int currentLength;
```

```
    public WrongLoginException(int currentLength) {
        super();
        this.currentLength = currentLength;
    }

    @Override
    public String getMessage() {
        return String.format("Your login is of incorrect length, expected < 20, given %d.",
                currentLength);
    }
}
```

Листинг 4: Формирование сообщения для класса исключения пароля

```
public static class WrongPasswordException extends RuntimeException {
    private int currentLength;
    private boolean matchConfirm;
    public WrongPasswordException(int currentLength, boolean matchConfirm) {
        super();
        this.currentLength = currentLength;
        this.matchConfirm = matchConfirm;
    }

    @Override
    public String getMessage() {
        boolean badlen = currentLength <= 20;
        return String.format("Your password is of %scorrect length%s %d. Password %smatch the
            confirmation.",
              ((badlen) ? "in" : ""),
              ((badlen) ? ", expected > 20, given" : ","),
              currentLength,
              (matchConfirm) ? "" : "doesn't ");
    }
}
```

Листинг 5: Создание тестовой среды

```
public static void main(String[] args) {
    String[][] credentials = {
          {"ivan", "1i2v3a4n5i6v7a8n91011", "1i2v3a4n5i6v7a8n91011"}, //correct
          {"1i2v3a4n5i6v7a8n91011", "", ""}, //wrong login length
          {"ivan", "1i2v3a4n5i6v7a8n91011", "1i2v3a4n5"}, //confirm mismatch
          {"ivan", "1i2v3a4n5", "1i2v3a4n5"},//wrong password length
          {"ivan", "1i2v3a4n5", "1i"} //wrong password length and confirm mismatch
    };
    for (int i = 0; i < credentials.length; i++) {
        try {
            System.out.println(checkCredentials(credentials[i][0], credentials[i][1],
                credentials[i][2]));
        } catch (WrongLoginException e) {
            e.printStackTrace();
        } catch (WrongPasswordException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Листинг 6: Метод проверки

```
public static boolean checkCredentials(String login, String password, String confirmPassword) {
    boolean conf = password.equals(confirmPassword);
    int llen = login.length();
    int plen = password.length();
```

```
    if (llen >= 20)
        throw new WrongLoginException(llen);
    else if (plen < 20 || !conf)
        throw new WrongPasswordException(plen, conf);
    else
        return true;
}
```

Задание 2

- **Ценность этапа** Написание наброска пет-проекта, повторение информации об ООП, работа с исключениями.
- **Тайминг** 35-40 минут.
- **Действия преподавателя**
  - Выдать задание студентам;
  - Подробно объяснить, что именно требуется от студентов, избегая упоминания конкретных языковых конструкций.
  - обратить внимание на порядок обработки исключений, повторная попытка купить товар может производиться только тогда, когда остальные параметры покупки уже проверены.
- **Задание**: Класс «Эмуляция интернет-магазина».
  1. Написать классы покупатель (ФИО, возраст, телефон), товар (название, цена) и заказ (объект покупатель, объект товар, целочисленное количество).
  2. Создать массив покупателей (инициализировать 2 элемента), массив товаров (инициализировать 5 элементов) и массив заказов (пустой на 5 элементов).
  3. Создать статический метод «совершить покупку» со строковыми параметрами, соответствующими полям объекта заказа. Метод должен вернуть объект заказа.
  4. Если в метод передан несуществующий покупатель – метод должен выбросить исключение `CustomerException`, если передан несуществующий товар, метод должен выбросить исключение `ProductException`, если было передано отрицательное или слишком больше значение количества (например, 100), метод должен выбросить исключение `AmountException`.
  5. Вызвать метод совершения покупки несколько раз таким образом, чтобы заполнить массив покупок возвращаемыми значениями. Обработать исключения следующим образом (в заданном порядке):
     - если был передан неверный товар – вывести в консоль сообщение об ошибке, не совершать данную покупку;
     - если было передано неверное количество – купить товар в количестве 1;
     - если был передан неверный пользователь – завершить работу приложения с исключением.
  6. Вывести в консоль итоговое количество совершённых покупок после выполнения основного кода приложения.

Вариант исполнения класса в приложении Б

Вариант маршрута решения задачи

Листинг 7: Создание базовых классов (идентично)

```
private static class Item {
   String name;
   int cost;

   public Item(String name, int cost) {
      this.name = name;
      this.cost = cost;
   }

   @Override
   public String toString() {
      return "Item{name='" + name + "', cost=" + cost + "}";
   }
}
```

Листинг 8: Создание пользовательских исключений

```
   public static class CustomerException extends RuntimeException {
      public CustomerException(String message) {
         super(message);
      }
   }
```

Листинг 9: Создание массивов

```
private final static Customer[] people = {
      new Customer("Ivan", 20, "+1-222-333-44-55"),
      new Customer("Petr", 30, "+2-333-444-55-66"),
};
private final static Item[] items = {
      new Item("Ball", 100),
      new Item("Sandwich", 1000),
      new Item("Table", 10000),
      new Item("Car", 100000),
      new Item("Rocket", 10000000)
};
private static Order[] orders = new Order[5];
```

Листинг 10: Описание тестовой среды

```
Object[][] info = {
      {people[0], items[0], 1}, //good
      {people[1], items[1], -1}, //bad amount -1
      {people[0], items[2], 150}, //bad amount >100
      {people[1], new Item("Flower", 10), 1}, //no item
      {new Customer("Fedor", 40, "+3-444-555-66-77"), items[3], 1}, //no customer
};
int capacity = 0;
int i = 0;
while (capacity != orders.length - 1 || i != info.length) {
   try {
      orders[capacity] = buy((Customer) info[i][0], (Item) info[i][1], (int) info[i][2]);
      capacity++;
   } catch (ProductException e) {
      e.printStackTrace();
   } catch (AmountException e) {
      orders[capacity++] = buy((Customer) info[i][0], (Item) info[i][1], 1);
   } catch (CustomerException e) {
      throw new RuntimeException(e);
   } finally {
      System.out.println("Orders made: " + capacity);
   }
   ++i;
```

```
}
```

Листинг 11: Написание и **отладка** основного метода

```
private static boolean isInArray(Object[] arr, Object o) {
   for (int i = 0; i < arr.length; i++)
      if (arr[i].equals(o)) return true;
   return false;
}

public static Order buy(Customer who, Item what, int howMuch) {
   if (!isInArray(people, who))
      throw new CustomerException("Unknown customer: " + who);
   if (!isInArray(items, what))
      throw new ProductException("Unknown item: " + what);
   if (howMuch < 0 || howMuch > 100)
      throw new AmountException("Incorrect amount: " + howMuch);
   return new Order(who, what, howMuch);
}
```

Домашнее задание

- **Ценность этапа** Задать задание для самостоятельного выполнения между занятиями.
- **Тайминг** 5-10 минут.
- **Действия преподавателя**
  - Пояснить студентам в каком виде выполнять и сдавать задания
  - Уточнить кто будет проверять работы (преподаватель или ревьювер)
  - Объяснить к кому обращаться за помощью и где искать подсказки
  - Объяснить где взять проект заготовки для дз
- **Задания**

5-25 мин Выполнить все задания семинара, если они не были решены, без ограничений по времени;

Все варианты решения приведены в тексте семинара выше

15 мин 1. В класс покупателя добавить перечисление с гендерами, добавить в сотрудника свойство «пол» со значением созданного перечисления. Добавить геттеры, сеттеры.

Листинг 12: Свойства сотрудника

```
enum Genders{MALE, FEMALE};

// ...
Genders gender;

public Employee(String name, String midName, String surName, String phone, String position,
    int salary, int birth, Genders gender) {
   // ...
   this.gender = gender;
}

public Genders getGender() {
   return gender;
}

public void setGender(Genders gender) {
```

```
16      this.gender = gender;
17  }
```

20-25 мин 2. Добавить в основную программу перечисление с праздниками (нет праздника, Новый Год, 8 марта, 23 февраля), написать метод, принимающий массив сотрудников, поздравляющий всех сотрудников с Новым Годом, женщин с 8 марта, а мужчин с 23 февраля, если сегодня соответствующий день.

Листинг 13: Праздники

```
1   enum Parties{NONE, NEW_YEAR, MARCH_8, FEB_23}
2   private static final Parties today = Parties.NONE;
3
4   private static void celebrate(Employee[] emp) {
5      for (int i = 0; i < emp.length; i++) {
6         switch (today) {
7            case NEW_YEAR:
8               System.out.println(emp[i].name + ", happy New Year!");
9               break;
10           case FEB_23:
11              if (emp[i].gender == Employee.Genders.MALE)
12                 System.out.println(emp[i].name + ", happy February 23rd!");
13              break;
14           case MARCH_8:
15              if (emp[i].gender == Employee.Genders.FEMALE)
16                 System.out.println(emp[i].name + ", happy march 8th!");
17              break;
18           default:
19              System.out.println(emp[i].name + ", celebrate this morning!");
20        }
21     }
22  }
```

Рефлексия и завершение семинара

- **Цель этапа:** Привести урок к логическому завершению, посмотреть что студентам удалось, что было сложно и над чем нужно еще поработать
- **Тайминг:** 5-10 минут
- **Действия преподавателя:**
  - Запросить обратную связь от студентов.
  - Подчеркните то, чему студенты научились на занятии.
  - Дайте рекомендации по решению заданий, если в этом есть необходимость
  - Дайте краткую обратную связь студентам.
  - Поделитесь ощущением от семинара.
  - Поблагодарите за проделанную работу.

Приложения

A. Домашнее задание 3

Листинг 14: Основная программа

```
package ru.gb.jcore;

import java.util.Arrays;

public class Exceptional {
   private static final class ColumnMismatchException extends RuntimeException {
      ColumnMismatchException(String message) {
         super("Columns exception: " + message);
      }
   }

   private static final class NumberIsNotNumberException extends RuntimeException {
      NumberIsNotNumberException(String message) {
         super("Not a number found: " + message);
      }
   }

   private static final class RowMismatchException extends RuntimeException {
      RowMismatchException(int expected, int current, String value) {
         super(String.format("Rows exception: expected %d rows. Received %d rows in '%s' string",
               expected, current, value));
      }
   }

   private static final String CORRECT_STRING= "1 3 1 2\n2 3 2 2\n5 6 7 1\n3 3 1 0";
   private static final String EXTRA_ROW_STRING= "1 3 1 2\n2 3 2 2\n5 6 7 1\n3 3 1 0\n1 2 3 4";
   private static final String EXTRA_COL_STRING= "1 3 1 2 1\n2 3 2 2 1\n5 6 7 1 1\n3 3 1 0 1";
   private static final String NO_ROW_STRING= "1 3 1 2\n2 3 2 2\n5 6 7 1";
   private static final String NO_COL_STRING= "1 3 1 2\n2 3 2 2\n5 6 7 1\n3 3 1";
   private static final String HAS_CHAR_STRING= "1 3 1 2\n2 3 2 2\n5 6 7 1\n3 3 1 A";

   private static final int MATRIX_ROWS= 4;
   private static final int MATRIX_COLS= 4;

   private static String[][] stringToMatrix(String value) {
      String[] rows = value.split("\n");
      if (rows.length !=MATRIX_ROWS)
         throw new RowMismatchException(MATRIX_ROWS, rows.length, value);

      String[][] result = new String[MATRIX_ROWS][];
      for (int i = 0; i < result.length; i++) {
         result[i] = rows[i].split(" ");
         if (result[i].length !=MATRIX_COLS)
            throw new ColumnMismatchException(result[i].length + ":\n" + value);
      }
      return result;
   }

   private static float calcMatrix(String[][] matrix) {
      float result = 0;
      int len = 0;
      for (int i = 0; i < matrix.length; i++) {
         for (int j = 0; j < matrix[i].length; j++) {
            try {
               result += Integer.parseInt(matrix[i][j]);
               ++len;
            } catch (NumberFormatException e) {
               throw new NumberIsNotNumberException(matrix[i][j]);
            }
```

```
        }
      }
      return result / len;
   }

   public static void main(String[] args) {
      try {
//         final String[][] matrix = stringToMatrix(CORRECT_STRING);
//         final String[][] matrix = stringToMatrix(NO_ROW_STRING);
//         final String[][] matrix = stringToMatrix(NO_COL_STRING);
         final String[][] matrix =stringToMatrix(HAS_CHAR_STRING);
         System.out.println(Arrays.deepToString(matrix));
         System.out.println("Half sum = " +calcMatrix(matrix));
      } catch (NumberIsNotNumberException exceptionObjectName) {
         System.out.println("A NumberFormatException is thrown: " + exceptionObjectName.getMessage());
      } catch (RowMismatchException | ColumnMismatchException superExceptionName) {
         System.out.println("A RuntimeException successor is thrown: " + superExceptionName.getMessage());
      }
   }
}
```

Б. Практическое задание 1

Листинг 15: Логин

```
package ru.gb.jcore;

public class SignInWorker {
   public static class WrongPasswordException extends RuntimeException {
      private int currentLength;
      private boolean matchConfirm;
      public WrongPasswordException(int currentLength, boolean matchConfirm) {
         super();
         this.currentLength = currentLength;
         this.matchConfirm = matchConfirm;
      }

      @Override
      public String getMessage() {
         boolean badlen = currentLength <= 20;
         return String.format("Your password is of %scorrect length%s %d. Password %smatch the
             confirmation.",
               ((badlen) ? "in" : ""),
               ((badlen) ? ", expected > 20, given" : ","),
               currentLength,
               (matchConfirm) ? "" : "doesn't ");
      }
   }

   public static class WrongLoginException extends RuntimeException {
      private int currentLength;
      public WrongLoginException(int currentLength) {
         super();
         this.currentLength = currentLength;
      }

      @Override
      public String getMessage() {
         return String.format("Your login is of incorrect length, expected < 20, given %d.",
               currentLength);
      }
   }

   public static boolean checkCredentials(String login, String password, String confirmPassword) {
```

```
        boolean conf = password.equals(confirmPassword);
        int llen = login.length();
        int plen = password.length();
        if (llen >= 20)
            throw new WrongLoginException(llen);
        else if (plen < 20 || !conf)
            throw new WrongPasswordException(plen, conf);
        else
            return true;
    }

    public static void main(String[] args) {
        String[][] credentials = {
                {"ivan", "1i2v3a4n5i6v7a8n91011", "1i2v3a4n5i6v7a8n91011"}, //correct
                {"1i2v3a4n5i6v7a8n91011", "", ""}, //wrong login length
                {"ivan", "1i2v3a4n5i6v7a8n91011", "1i2v3a4n5"}, //confirm mismatch
                {"ivan", "1i2v3a4n5", "1i2v3a4n5"},//wrong password length
                {"ivan", "1i2v3a4n5", "1i"} //wrong password length and confirm mismatch
        };
        for (int i = 0; i < credentials.length; i++) {
            try {
                System.out.println(checkCredentials(credentials[i][0], credentials[i][1], credentials[i][2]));
            } catch (WrongLoginException e) {
                e.printStackTrace();
            } catch (WrongPasswordException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

B. Практическое задание 2

Листинг 16: Магазин

```
package ru.gb.jcore;

public class Shop {
    /**
     * Вызватьметодсовершенияпокупкинесколькоразтакимобразом       ,
     * чтобызаполнитьмассивпокуповозвращаемымизначениями      .
     * Обработатьисключениясследующимобразомв     ( заданномпорядке ):
     * */
    private static class Customer {
        String name;
        int age;
        String phone;

        public Customer(String name, int age, String phone) {
            this.name = name;
            this.age = age;
            this.phone = phone;
        }

        @Override
        public String toString() {
            return "Customer{name='" + name + '\'' +
                    ", age=" + age + ", phone='" + phone + "'}";
        }
    }
    private static class Item {
        String name;
        int cost;

```

```
30          public Item(String name, int cost) {
31              this.name = name;
32              this.cost = cost;
33          }
34
35          @Override
36          public String toString() {
37              return "Item{name='" + name + "', cost=" + cost + "}";
38          }
39      }
40
41      private static class Order {
42          Customer customer;
43          Item item;
44          int amount;
45
46          public Order(Customer customer, Item item, int amount) {
47              this.customer = customer;
48              this.item = item;
49              this.amount = amount;
50          }
51
52          @Override
53          public String toString() {
54              return "Order{customer=" + customer +
55                     ", item=" + item + ", amount=" + amount + "}";
56          }
57      }
58
59      public static class CustomerException extends RuntimeException {
60          public CustomerException(String message) { super(message); }
61      }
62      public static class ProductException extends RuntimeException {
63          public ProductException(String message) { super(message); }
64      }
65      public static class AmountException extends RuntimeException {
66          public AmountException(String message) { super(message); }
67      }
68
69      private final static Customer[] people = {
70              new Customer("Ivan", 20, "+1-222-333-44-55"),
71              new Customer("Petr", 30, "+2-333-444-55-66"),
72      };
73      private final static Item[] items = {
74              new Item("Ball", 100),
75              new Item("Sandwich", 1000),
76              new Item("Table", 10000),
77              new Item("Car", 100000),
78              new Item("Rocket", 10000000)
79      };
80      private static Order[] orders = new Order[5];
81
82      private static boolean isInArray(Object[] arr, Object o) {
83          for (int i = 0; i < arr.length; i++)
84              if (arr[i].equals(o)) return true;
85          return false;
86      }
87
88      public static Order buy(Customer who, Item what, int howMuch) {
89          if (!isInArray(people, who))
90              throw new CustomerException("Unknown customer: " + who);
91          if (!isInArray(items, what))
92              throw new ProductException("Unknown item: " + what);
93          if (howMuch < 0 || howMuch > 100)
94              throw new AmountException("Incorrect amount: " + howMuch);
95          return new Order(who, what, howMuch);
```

```
    }

    public static void main(String[] args) {
        Object[][] info = {
                {people[0], items[0], 1}, //good
                {people[1], items[1], -1}, //bad amount -1
                {people[0], items[2], 150}, //bad amount >100
                {people[1], new Item("Flower", 10), 1}, //no item
                {new Customer("Fedor", 40, "+3-444-555-66-77"), items[3], 1}, //no customer
        };
        int capacity = 0;
        int i = 0;
        while (capacity != orders.length - 1 || i != info.length) {
            try {
                orders[capacity] = buy((Customer) info[i][0], (Item) info[i][1], (int) info[i][2]);
                capacity++;
            } catch (ProductException e) {
                e.printStackTrace();
            } catch (AmountException e) {
                orders[capacity++] = buy((Customer) info[i][0], (Item) info[i][1], 1);
            } catch (CustomerException e) {
                throw new RuntimeException(e);
            } finally {
                System.out.println("Orders made: " + capacity);
            }
            ++i;
        }
    }
}
```