

## 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, озвучивает цели и планы на семинар. Важно упомянуть, что выполнение домашних заданий с лекции является, фактически, подготовкой к семинару
Quiz	10	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	25-28	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	33	Преподаватель предлагает студентам перерыв на 5 минут (студенты голосуют)
Задание 2	40	29-32	Сквозное задание, состоящее из объяснений в 6 пунктах, обязательных к исполнению. Всё задание выдаётся сразу целиком и является неделимым.
Домашнее задание	5	45-46	Объясните домашнее задание, подведите итоги урока
Рефлексия	10	47-48	Преподаватель запрашивает обратную связь
Длительность	125		

## 4.4. Подробности

### 4.4.1. 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.

### 4.4.2. 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

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

### 4.4.3. 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

```
1 private static final class ColumnMismatchException extends RuntimeException {
2     ColumnMismatchException(String message) {
3         super("Columns exception: " + message);
4     }
5 }
6
7 private static final class NumberIsNotNumberException extends RuntimeException {
8     NumberIsNotNumberException(String message) {
9         super("Not a number found: " + message);
10    }
11 }
```

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

Листинг 2: Составное сообщение

```
1 private static final class RowMismatchException extends RuntimeException {
2     RowMismatchException(int expected, int current, String value) {
3         super(String.format("Rows exception: expected %d rows. Received %d rows in '%s' string",
4             expected, current, value));
5     }
6 }
```

- 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

#### 4.4.4. Вопросы и ответы

- **Ценность этапа** Задать задание для самостоятельного выполнения между занятиями.
- **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.

#### 4.4.5. 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 `confirmPassword` 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;
```

```

3 public WrongLoginException(int currentLength) {
4     super();
5     this.currentLength = currentLength;
6 }
7
8 @Override
9 public String getMessage() {
10    return String.format("Your login is of incorrect length, expected < 20, given %d.",
11        currentLength);
12 }
13 }

```

Листинг 4: Формирование сообщения для класса исключения пароля

```

1 public static class WrongPasswordException extends RuntimeException {
2     private int currentLength;
3     private boolean matchConfirm;
4     public WrongPasswordException(int currentLength, boolean matchConfirm) {
5         super();
6         this.currentLength = currentLength;
7         this.matchConfirm = matchConfirm;
8     }
9
10    @Override
11    public String getMessage() {
12        boolean badlen = currentLength <= 20;
13        return String.format("Your password is of %scorrect length%s %d. Password %smatch the
14            confirmation.",
15            ((badlen) ? "in" : ""),
16            ((badlen) ? ", expected > 20, given" : ","),
17            currentLength,
18            (matchConfirm) ? "" : "doesn't ");
19    }
20 }

```

Листинг 5: Создание тестовой среды

```

1 public static void main(String[] args) {
2     String[][] credentials = {
3         {"ivan", "1i2v3a4n5i6v7a8n91011", "1i2v3a4n5i6v7a8n91011"}, //correct
4         {"1i2v3a4n5i6v7a8n91011", "", ""}, //wrong login length
5         {"ivan", "1i2v3a4n5i6v7a8n91011", "1i2v3a4n5"}, //confirm mismatch
6         {"ivan", "1i2v3a4n5", "1i2v3a4n5"}, //wrong password length
7         {"ivan", "1i2v3a4n5", "1i"} //wrong password length and confirm mismatch
8     };
9     for (int i = 0; i < credentials.length; i++) {
10        try {
11            System.out.println(checkCredentials(credentials[i][0], credentials[i][1],
12                credentials[i][2]));
13        } catch (WrongLoginException e) {
14            e.printStackTrace();
15        } catch (WrongPasswordException e) {
16            System.out.println(e.getMessage());
17        }
18    }
19 }

```

Листинг 6: Метод проверки

```

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

```

```

5     if (llen >= 20)
6         throw new WrongLoginException(llen);
7     else if (plen < 20 || !conf)
8         throw new WrongPasswordException(plen, conf);
9     else
10        return true;
11 }

```

#### 4.4.6. 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 большое 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)

```

1 private static class Item {
2     String name;
3     int cost;
4
5     public Item(String name, int cost) {
6         this.name = name;
7         this.cost = cost;
8     }
9
10    @Override
11    public String toString() {
12        return "Item{name='" + name + "', cost=" + cost + "}";
13    }
14 }

```

### Листинг 8: Создание пользовательских исключений

```

1 public static class CustomerException extends RuntimeException {
2     public CustomerException(String message) {
3         super(message);
4     }
5 }

```

### Листинг 9: Создание массивов

```

1 private final static Customer[] people = {
2     new Customer("Ivan", 20, "+1-222-333-44-55"),
3     new Customer("Petr", 30, "+2-333-444-55-66"),
4 };
5 private final static Item[] items = {
6     new Item("Ball", 100),
7     new Item("Sandwich", 1000),
8     new Item("Table", 10000),
9     new Item("Car", 100000),
10    new Item("Rocket", 10000000)
11 };
12 private static Order[] orders = new Order[5];

```

### Листинг 10: Описание тестовой среды

```

1 Object[][] info = {
2     {people[0], items[0], 1}, //good
3     {people[1], items[1], -1}, //bad amount -1
4     {people[0], items[2], 150}, //bad amount >100
5     {people[1], new Item("Flower", 10), 1}, //no item
6     {new Customer("Fedor", 40, "+3-444-555-66-77"), items[3], 1}, //no customer
7 };
8 int capacity = 0;
9 int i = 0;
10 while (capacity != orders.length - 1 || i != info.length) {
11     try {
12         orders[capacity] = buy((Customer) info[i][0], (Item) info[i][1], (int) info[i][2]);
13         capacity++;
14     } catch (ProductException e) {
15         e.printStackTrace();
16     } catch (AmountException e) {
17         orders[capacity++] = buy((Customer) info[i][0], (Item) info[i][1], 1);
18     } catch (CustomerException e) {
19         throw new RuntimeException(e);
20     } finally {
21         System.out.println("Orders made: " + capacity);
22     }
23     ++i;

```

## Листинг 11: Написание и отладка основного метода

```

1 private static boolean isInArray(Object[] arr, Object o) {
2     for (int i = 0; i < arr.length; i++)
3         if (arr[i].equals(o)) return true;
4     return false;
5 }
6
7 public static Order buy(Customer who, Item what, int howMuch) {
8     if (!isInArray(people, who))
9         throw new CustomerException("Unknown customer: " + who);
10    if (!isInArray(items, what))
11        throw new ProductException("Unknown item: " + what);
12    if (howMuch < 0 || howMuch > 100)
13        throw new AmountException("Incorrect amount: " + howMuch);
14    return new Order(who, what, howMuch);
15 }

```

## 4.4.7. 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 ревьюер)
- 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

```

1 enum Genders{MALE, FEMALE};
2
3 // ...
4 Genders gender;
5
6 public Employee(String name, String midName, String surName, String phone, String position,
7     int salary, int birth, Genders gender) {
8     // ...
9     this.gender = gender;
10 }
11 public Genders getGender() {
12     return gender;
13 }
14
15 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 }
```

#### 4.4.8. 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

## А. Домашнее задание 3

Листинг 14: Основная программа

```
1 package ru.gb.jcore;
2
3 import java.util.Arrays;
4
5 public class Exceptional {
6     private static final class ColumnMismatchException extends RuntimeException {
7         ColumnMismatchException(String message) {
8             super("Columns exception: " + message);
9         }
10    }
11
12    private static final class NumberIsNotNumberException extends RuntimeException {
13        NumberIsNotNumberException(String message) {
14            super("Not a number found: " + message);
15        }
16    }
17
18    private static final class RowMismatchException extends RuntimeException {
19        RowMismatchException(int expected, int current, String value) {
20            super(String.format("Rows exception: expected %d rows. Received %d rows in '%s' string",
21                expected, current, value));
22        }
23    }
24
25    private static final String CORRECT_STRING= "1 3 1 2\n2 3 2 2\n5 6 7 1\n3 3 1 0";
26    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";
27    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";
28    private static final String NO_ROW_STRING= "1 3 1 2\n2 3 2 2\n5 6 7 1";
29    private static final String NO_COL_STRING= "1 3 1 2\n2 3 2 2\n5 6 7 1\n3 3 1";
30    private static final String HAS_CHAR_STRING= "1 3 1 2\n2 3 2 2\n5 6 7 1\n3 3 1 A";
31
32    private static final int MATRIX_ROWS= 4;
33    private static final int MATRIX_COLS= 4;
34
35    private static String[][] stringToMatrix(String value) {
36        String[] rows = value.split("\n");
37        if (rows.length !=MATRIX_ROWS)
38            throw new RowMismatchException(MATRIX_ROWS, rows.length, value);
39
40        String[][] result = new String[MATRIX_ROWS][];
41        for (int i = 0; i < result.length; i++) {
42            result[i] = rows[i].split(" ");
43            if (result[i].length !=MATRIX_COLS)
44                throw new ColumnMismatchException(result[i].length + ":\n" + value);
45        }
46        return result;
47    }
48
49    private static float calcMatrix(String[][] matrix) {
50        float result = 0;
51        int len = 0;
52        for (int i = 0; i < matrix.length; i++) {
53            for (int j = 0; j < matrix[i].length; j++) {
54                try {
55                    result += Integer.parseInt(matrix[i][j]);
56                    ++len;
57                } catch (NumberFormatException e) {
58                    throw new NumberIsNotNumberException(matrix[i][j]);
59                }
60            }
61        }
62        return result/len;
63    }
64}
```

```

59     }
60   }
61 }
62 return result / len;
63 }
64
65 public static void main(String[] args) {
66   try {
67     // final String[][] matrix = stringToMatrix(CORRECT_STRING);
68     // final String[][] matrix = stringToMatrix(NO_ROW_STRING);
69     // final String[][] matrix = stringToMatrix(NO_COL_STRING);
70     final String[][] matrix = stringToMatrix(HAS_CHAR_STRING);
71     System.out.println(Arrays.deepToString(matrix));
72     System.out.println("Half sum = " + calcMatrix(matrix));
73   } catch (NumberFormatException exceptionObjectName) {
74     System.out.println("A NumberFormatException is thrown: " + exceptionObjectName.getMessage());
75   } catch (RowMismatchException | ColumnMismatchException superExceptionName) {
76     System.out.println("A RuntimeException successor is thrown: " + superExceptionName.getMessage());
77   }
78 }
79 }

```

## Б. Практическое задание 1

Листинг 15: Логин

```

1 package ru.gb.jcore;
2
3 public class SignInWorker {
4   public static class WrongPasswordException extends RuntimeException {
5     private int currentLength;
6     private boolean matchConfirm;
7     public WrongPasswordException(int currentLength, boolean matchConfirm) {
8       super();
9       this.currentLength = currentLength;
10      this.matchConfirm = matchConfirm;
11    }
12
13    @Override
14    public String getMessage() {
15      boolean badlen = currentLength <= 20;
16      return String.format("Your password is of %scorrect length%s %d. Password %smatch the
17        confirmation.",
18        ((badlen) ? "in" : ""),
19        ((badlen) ? ", expected > 20, given" : ","),
20        currentLength,
21        (matchConfirm) ? "" : "doesn't ");
22    }
23  }
24
25  public static class WrongLoginException extends RuntimeException {
26    private int currentLength;
27    public WrongLoginException(int currentLength) {
28      super();
29      this.currentLength = currentLength;
30    }
31
32    @Override
33    public String getMessage() {
34      return String.format("Your login is of incorrect length, expected < 20, given %d.",
35        currentLength);
36    }
37  }

```

```

37
38 public static boolean checkCredentials(String login, String password, String confirmPassword) {
39     boolean conf = password.equals(confirmPassword);
40     int llen = login.length();
41     int plen = password.length();
42     if (llen >= 20)
43         throw new WrongLoginException(llen);
44     else if (plen < 20 || !conf)
45         throw new WrongPasswordException(plen, conf);
46     else
47         return true;
48 }
49
50 public static void main(String[] args) {
51     String[][] credentials = {
52         {"ivan", "1i2v3a4n5i6v7a8n9i011", "1i2v3a4n5i6v7a8n9i011"}, //correct
53         {"1i2v3a4n5i6v7a8n9i011", "", ""}, //wrong login length
54         {"ivan", "1i2v3a4n5i6v7a8n9i011", "1i2v3a4n5"}, //confirm mismatch
55         {"ivan", "1i2v3a4n5", "1i2v3a4n5"}, //wrong password length
56         {"ivan", "1i2v3a4n5", "1i"} //wrong password length and confirm mismatch
57     };
58     for (int i = 0; i < credentials.length; i++) {
59         try {
60             System.out.println(checkCredentials(credentials[i][0], credentials[i][1], credentials[i][2]));
61         } catch (WrongLoginException e) {
62             e.printStackTrace();
63         } catch (WrongPasswordException e) {
64             System.out.println(e.getMessage());
65         }
66     }
67 }
68 }

```

## В. Практическое задание 2

Листинг 16: Магазин

```

1 package ru.gb.jcore;
2
3 public class Shop {
4     /**
5      * Вызвать метод совершения покупки несколько раз таким образом
6      * чтобы заполнить массив покупок возвращаемыми значениями
7      * Обработать исключения следующим образом ( заданном порядке ):
8      * */
9     private static class Customer {
10         String name;
11         int age;
12         String phone;
13
14         public Customer(String name, int age, String phone) {
15             this.name = name;
16             this.age = age;
17             this.phone = phone;
18         }
19
20         @Override
21         public String toString() {
22             return "Customer{name='" + name + '\'' +
23                 ", age=" + age + ", phone='" + phone + "'}";
24         }
25     }
26     private static class Item {

```

```

27     String name;
28     int cost;
29
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);
96 }
97
98 public static void main(String[] args) {
99     Object[][] info = {
100         {people[0], items[0], 1}, //good
101         {people[1], items[1], -1}, //bad amount -1
102         {people[0], items[2], 150}, //bad amount >100
103         {people[1], new Item("Flower", 10), 1}, //no item
104         {new Customer("Fedor", 40, "+3-444-555-66-77"), items[3], 1}, //no customer
105     };
106     int capacity = 0;
107     int i = 0;
108     while (capacity != orders.length - 1 || i != info.length) {
109         try {
110             orders[capacity] = buy((Customer) info[i][0], (Item) info[i][1], (int) info[i][2]);
111             capacity++;
112         } catch (ProductException e) {
113             e.printStackTrace();
114         } catch (AmountException e) {
115             orders[capacity++] = buy((Customer) info[i][0], (Item) info[i][1], 1);
116         } catch (CustomerException e) {
117             throw new RuntimeException(e);
118         } finally {
119             System.out.println("Orders made: " + capacity);
120         }
121         ++i;
122     }
123 }
124 }

```