Практика 6

1. Напишите код, чтобы получить **стек-трейс** длиной **10 вызовов**. Метод main изменять нельзя.

```
public class Solution {
	public static void main(String[] args) {
		int stackTraceLength = method1().length - method10().length + 1;
	}

	public static StackTraceElement[] method1() {
		return method2();
	}

	public static StackTraceElement[] method2() {
		//напишите тут ваш код
	}

	public static StackTraceElement[] method3() {
		//напишите тут ваш код
	}

	public static StackTraceElement[] method4() {
		//напишите тут ваш код
	}

	public static StackTraceElement[] method5() {
		//напишите тут ваш код
	}

	public static StackTraceElement[] method6() {
		//напишите тут ваш код
	}

	public static StackTraceElement[] method7() {
		//напишите тут ваш код
	}

	public static StackTraceElement[] method8() {
		//напишите тут ваш код
	}

	public static StackTraceElement[] method9() {
		return method10();
	}

	public static StackTraceElement[] method10() {
		return Thread.currentThread().getStackTrace();
	}
}
```

2. Написать **метод**, который возвращает **результат - глубину** его **стек-трейса** - **количество методов** в нем (количество элементов в списке). Это же число метод должен выводить на экран.

```
public class Solution {
	public static void main(String[] args) {
		int deep = getStackTraceDepth();
	}

	public static int getStackTraceDepth() {
		//напишите тут ваш код
	}
}
```

3. Реализовать метод **log**. Он должен выводить на экран **имя класса** и **имя метода** (в котором вызывается метод **log**), а также переданное **сообщение**.
Имя класса, **имя метода** и **сообщение** разделить двоеточием с пробелом.
Пример вывода:
csb.Solution: main: In main method

```
public class Solution {
	public static void main(String[] args) {
		log("In main method");
	}

	public static void log(String s) {
		//напишите тут ваш код
	}
}
```

4. **Перехватить** исключение (и вывести его на экран), указав его тип, возникающее при выполнении кода:
```
int[] m = new int[2];
m[8] = 5;
```

5. **Перехватить** исключение (и вывести его на экран, указав его тип), возникающее при выполнении кода:
```
ArrayList<String> list = new ArrayList<String>();
String s = list.get(18);
```

6. **Перехватить** исключение (и вывести его на экран, указав его тип), возникающее при выполнении кода:
```
HashMap map = new HashMap(null);
map.put(null, null);
map.remove(null);
```

7. **Перехватить** исключение (и вывести его на экран, указав его тип), возникающее при выполнении кода:
```
int num=Integer.parseInt("XYZ");
System.out.println(num);
```

8. Есть метод, который выбрасывает два исключения, унаследованные от Exception, и два унаследованных от RuntimeException:NullPointerException, ArithmeticException, FileNotFoundException, URISyntaxException. Нужно перехватить **NullPointerException** и **FileNotFoundException**, но не перехватывать ArithmeticException и URISyntaxException. Метод method1 не изменять.

```
public class Solution {
    public static void main(String[] args) throws Exception {
        //напишите тут ваш код
        method1();

        //напишите тут ваш код
    }

    public static void method1() throws NullPointerException, ArithmeticException, FileNotFoundException, URISyntaxException {
        int i = (int) (Math.random() * 4);
        if (i == 0) {
            throw new NullPointerException();
        } else if (i == 1) {
            throw new ArithmeticException();
        } else if (i == 2) {
            throw new FileNotFoundException();
        } else if (i == 3) {
            throw new URISyntaxException("", "");
        }
    }
}
```

9. 1. Есть три исключения последовательно унаследованные от **Exception**:
   2. class Exception1 extends Exception
   3. class Exception2 extends Exception1
   4. class Exception3 extends Exception2
   5. Есть **метод**, который описан так:
   public static void method1() throws Exception1, Exception2, Exception3
   6. Напишите catch, который перехватит все три Exception1, Exception2 и Exception3

```
public class Solution {
    public static void main(String[] args) throws Exception {
        //напишите тут ваш код
        method1();
        //напишите тут ваш код
    }

    public static void method1() throws Exception1, Exception2, Exception3 {
        int i = (int) (Math.random() * 3);
        if (i == 0) {
            throw new Exception1();
        } else if (i == 1) {
            throw new Exception2();
        } else if (i == 2) {
            throw new Exception3();
        }
    }
}

class Exception1 extends Exception {
}

class Exception2 extends Exception1 {
}

class Exception3 extends Exception2 {
}
```

10. 1. Разберитесь, какие исключения бросает метод BEAN.methodThrowExceptions.
2. Метод handleExceptions должен вызывать метод BEAN.methodThrowExceptions и обрабатывать исключения:
2.1. если возникло исключение FileSystemException, то логировать его (вызвать метод **BEAN.log**) и пробросить дальше
2.2. если возникло исключение CharConversionException или любое другое IOException, то только логировать его (вызвать метод **BEAN.log**)
3. Добавьте в объявление метода handleExceptions класс исключения, которое вы пробрасываете в п.2.1.
4. В методе main обработайте оставшееся исключение - логируйте его. Используйте try..catch
Подсказка: Если вы захватили исключение MyException, которое не хотели захватывать, его можно пробросить дальше кодом вида:
catch (MyException e) {
 throw e;
}

```
public class Solution {
    public static StatelessBean BEAN = new StatelessBean();

    public static void main(String[] args) {
        handleExceptions();
    }

    public static void handleExceptions() {
        BEAN.methodThrowExceptions();
    }

    public static class StatelessBean {
        public void log(Exception exception) {
            System.out.println(exception.getMessage() + ", " + exception.getClass().getSimpleName());
        }

        public void methodThrowExceptions() throws CharConversionException, FileSystemException, IOException {
            int i = (int) (Math.random() * 3);
            if (i == 0) {
                throw new CharConversionException();
            } else if (i == 1) {
                throw new FileSystemException("");
            } else if (i == 2) {
                throw new IOException();
            }
        }
    }
}
```

11. В методе handleExceptions обработайте все checked исключения. Нужно вывести на экран возникшее checked исключение. Можно использовать только один блок try..catch

```
public class Solution {
	public static void main(String[] args) {
		handleExceptions(new Solution());

	}

	public static void handleExceptions(Solution obj) {
		obj.method1();
		obj.method2();
		obj.method3();
	}

	public void method1() throws IOException {
		throw new IOException();
	}

	public void method2() throws NoSuchFieldException {
		throw new NoSuchFieldException();
	}

	public void method3() throws RemoteException {
		throw new RemoteException();
	}
}
```

12. В методе handleExceptions обработайте все unchecked исключения.
    Нужно вывести стек-трейс возникшего исключения используя метод printStack.
    Можно использовать только один блок try..catch

```
public class Solution {
	public static void main(String[] args) {
		handleExceptions(new Solution());
	}

	public static void handleExceptions(Solution obj) {
		obj.method1();
		obj.method2();
		obj.method3();
	}

	public static void printStack(Throwable throwable) {
		System.out.println(throwable);
		for (StackTraceElement element : throwable.getStackTrace()) {
			System.out.println(element);
		}
	}

	public void method1() {
		throw new NullPointerException();
	}

	public void method2() {
		throw new IndexOutOfBoundsException();
	}

	public void method3() {
		throw new NumberFormatException();
	}
}
```

13. Есть четыре класса **MyException**, **MyException2**, **MyException3**, **MyException4**.
Унаследуйте **классы** так, чтобы у вас появилось любые два checked исключения и любые два unchecked исключения.
Подсказка:
Изучите внимательно классы **Exception1**, **Exception2**, **Exception3** из второй задачи этого блока.

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

    static class MyException {
    }

    static class MyException2 {
    }

    static class MyException3 {
    }

    static class MyException4 {
    }
}
```

14. Создайте метод `public static void divideByZero`, в котором поделите **любое число на ноль** и выведите на экран результат деления. Оберните вызов метода `divideByZero` в `try..catch`. Выведите **стек-трейс** исключения используя метод `exception.printStackTrace()`

```
public class Solution {
	public static void main(String[] args) {
		divideByZero();
	}
}
```

15. Написать в цикле обратный отсчёт от **10** до **0**. Для задержки использовать `Thread.sleep(100);` Обернуть вызов `sleep` в `try..catch.`

```
public class Solution {
	public static void main(String[] args) {
		for (int i = 10; i >= 0; i--) {
			System.out.println(i);
			//напишите тут ваш код
		}
	}
}
```

16. Написать программу, которая будет вводить **числа** с клавиатуры. Код по чтению **чисел** с клавиатуры должен быть в методе `readData`. Код внутри `readData` обернуть в `try..catch`. Если пользователь ввёл **какой-то текст**, вместо ввода **числа**, то метод должен **перехватить исключение** и вывести на экран все ранее **введенные числа** в качестве результата. Числа выводить с новой строки сохраняя порядок ввода.

```
public class Solution {
	public static void main(String[] args) {
		readData();
	}

	public static void readData() {
		//напишите тут ваш код
	}
}
```

17. Ввести с клавиатуры дату в формате "**2013-08-18**"
Вывести на экран введенную дату в виде "**AUG 18, 2013**". Воспользоваться объектом Date и SimpleDateFormat.

18. Расставьте модификаторы **static** так, чтобы пример скомпилировался. (В классе должно быть 3 статических поля.)

```
public class Solution {
	public int A = 5;
	public int B = 2 * A;
	public int C = A * B;
	public int D = A * B;

	public static void main(String[] args) {
		Solution room = new Solution();
		room.A = 5;
		Solution.D = 5;
	}

	public int getA() {
		return A;
	}
}
```

19. Создать список, элементами которого будут **массивы чисел**. Добавить в список пять **объектов-массивов** длиной **5, 2, 4, 7, 0** соответственно. Заполнить массивы **любыми данными** и вывести их на экран.

```
public class Solution {
	public static void main(String[] args) {
		ArrayList<int[]> list = createList();
		printList(list);
	}

	public static ArrayList<int[]> createList() {
		//напишите тут ваш код
	}

	public static void printList(ArrayList<int[]> list) {
		for (int[] array : list) {
			for (int x : array) {
				System.out.println(x);
			}
		}
	}
}
```

20. Написать программу, которая вводит с клавиатуры **строку текста**.
Программа должна вывести на экран две строки:
1. **первая строка** содержит только **гласные** буквы из введённой строки.
2. **вторая** - только **согласные** буквы и **знаки препинания** из введённой строки.
Буквы соединять пробелом, каждая строка должна заканчиваться пробелом.
Пример ввода: Мама мыла раму.
Пример вывода:
а а ы а а у
М м м л р м .

```
public class Solution {
    public static char[] vowels = new char[]{'а', 'я', 'у', 'ю', 'и', 'ы', 'э', 'е', 'о', 'ё'};

    public static void main(String[] args) throws Exception {
        //напишите тут ваш код
    }

    // метод проверяет, гласная ли буква
    public static boolean isVowel(char c) {
        c = Character.toLowerCase(c);  // приводим символ в нижний регистр - от заглавных к строчным буквам
        for (char d : vowels) {  // ищем среди массива гласных
            if (c == d) {
                return true;
            }
        }
        return false;
    }
}
```

21. Программа вводит два имени файла. И копирует первый файл на место заданное вторым именем.

```
public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

        String sourceFileName = reader.readLine();
        String destinationFileName = reader.readLine();

        InputStream fileInputStream = getOutputStream(destinationFileName);
        InputStream fileOutputStream = getOutputStream(destinationFileName);

        int count = 0;
        while (fileInputStream.available() > 0) ;
        {
            int data = fileInputStream.read();
            fileOutputStream.write(data);
            count++;
        }

        System.out.println("Скопировано байт " + count);

        fileInputStream.close();
        fileOutputStream.close();
    }

    public static InputStream getInputStream(String fileName) throws IOException {
        return new FileInputStream(fileName);
    }

    public static OutputStream getOutputStream(String fileName) throws IOException {
        return new FileOutputStream(fileName);
    }
}
```

22. Задача: Программа вводит **два имени файла**. И копирует первый файл на место, заданное вторым именем.
Новая задача: Программа вводит **два имени файла**. И копирует первый файл на место, заданное вторым именем.
Если файла (который нужно копировать) с указанным именем не существует, то программа должна вывести надпись "**Файл не существует.**" и еще один раз **прочитать имя файла с консоли**, а уже потом считывать файл для записи.

```
public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

        String sourceFileName = reader.readLine();
        String destinationFileName = reader.readLine();

        InputStream fileInputStream = getInputStream(sourceFileName);
        OutputStream fileOutputStream = getOutputStream(destinationFileName);

        while (fileInputStream.available() > 0) {
            int data = fileInputStream.read();
            fileOutputStream.write(data);
        }

        fileInputStream.close();
        fileOutputStream.close();
    }

    public static InputStream getInputStream(String fileName) throws IOException {
        return new FileInputStream(fileName);
    }

    public static OutputStream getOutputStream(String fileName) throws IOException {
        return new FileOutputStream(fileName);
    }
}
```

23. 1. Есть пять классов: красная шапочка, бабушка, пирожок, дровосек, волк.
2. У каждого класса есть **2** поля: убил (killed ArrayList) и съел (ate ArrayList).
3. Необходимые **объекты** созданы (hood, grandmother, ...).
4. Расставь правильно связи, кто кого **съел** и **убил**, чтобы получилась логика сказки "**Красная Шапочка**".
PS: пирожки никто не ел. Их только несли. Волк чуток поел. А его потом убили.

```
public class Solution {
    public static LittleRedRidingHood hood = new LittleRedRidingHood();
    public static Grandmother grandmother = new Grandmother();
    public static Patty patty = new Patty();
    public static Woodman woodman = new Woodman();
    public static Wolf wolf = new Wolf();

    public static void main(String[] args) {
        // напишите тут ваш код
    }

    // Красная шапочка
    public static class LittleRedRidingHood extends StoryItem {
    }

    // Бабушка
    public static class Grandmother extends StoryItem {
    }

    // Пирожок
    public static class Patty extends StoryItem {
    }

    // Дровосек
    public static class Woodman extends StoryItem {
    }

    // Волк
    public static class Wolf extends StoryItem {
    }

    public static abstract class StoryItem {
        public ArrayList<StoryItem> killed = new ArrayList<>();
        public ArrayList<StoryItem> ate = new ArrayList<>();
    }
}
```

24. Есть класс кот - **Cat**, с полем "**имя**" (String).
Создать словарь **Map**<String, Cat> и добавить туда **10** котов в виде "**Имя**"-"**Кот**".
Получить из **Map** множество(Set) всех котов и вывести его на экран.

```
public class Solution {
	public static void main(String[] args) {
		Map<String, Cat> map = createMap();
		Set<Cat> set = convertMapToSet(map);
		printCatSet(set);
	}

	public static Map<String, Cat> createMap() {
		//напишите тут ваш код
	}

	public static Set<Cat> convertMapToSet(Map<String, Cat> map) {
		//напишите тут ваш код
	}

	public static void printCatSet(Set<Cat> set) {
		for (Cat cat : set) {
			System.out.println(cat);
		}
	}

	public static class Cat {
		private String name;

		public Cat(String name) {
			this.name = name;
		}

		public String toString() {
			return "Cat " + this.name;
		}
	}
}
```

25. Задача: Пользователь вводит с клавиатуры **список слов** (и чисел). **Слова** вывести **в возрастающем** порядке, **числа** - **в убывающем**.

- Пример вывода:
  Арбуз
  22
  Боб
  3
  Вишня
  1
  0
  Яблоко
- Пример ввода:
  Вишня
  1
  Боб
  3
  Яблоко
  22
  0
  Арбуз

```
public class Solution {
	public static void main(String[] args) throws Exception {
		BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
		ArrayList<String> list = new ArrayList<>();
		while (true) {
			String s = reader.readLine();
			if (s.isEmpty()) {
				break;
			}
			list.add(s);
		}
		String[] array = list.toArray(new String[0]);
		sort(array);
		for (String x : array) {
			System.out.println(x);
		}
	}

	public static void sort(String[] array) {
		// напишите тут ваш код
	}
	// Метод для сравнения строк: 'a' больше чем 'b'
	public static boolean isGreaterThan(String a, String b) {
		return a.compareTo(b) > 0;
	}
	// Переданная строка - это число?
	public static boolean isNumber(String s) {
		if (s.length() == 0) {
			return false;
		}
		char[] chars = s.toCharArray();
		for (int i = 0; i < chars.length; i++) {
			char c = chars[i];
			if ((i != 0 && c == '-') // Строка содержит '-'
					|| (!Character.isDigit(c) && c != '-') // или не цифра и не начинается с '-'
					|| (chars.length == 1 && c == '-')) // или одиночный '-'
			{
				return false;
			}
		}
		return true;
	}
}
```

Следующие задачи необходимо решать через аргументы, а не BufferedReader.

26. Написать функцию, которая будет выбирать меньшее число из нескольких введённых.

27. Написать функцию, которая будет считывать вводимые параметры, и будет выводить сумму чётных индексов позиций введённых чисел (0+2+4+6..) и разность нечётных индексов позиций введённых чисел (1-3-5-7..)

28. Остановить предыдущую функцию, когда чётные станут больше 100 или нечётные станут меньше 0.

29. Написать функцию, в которую передаём дату рождения, а она показывает сколько до ближайших (проверка в обе стороны). Если день рождения был вчера, то назад = 1, а вперед 364. (формат 30.07.2019)