Практика 5

1. Создать коллекцию **Set (реализация HashSet)** с типом элементов String.
   Добавить в неё **10** строк:
   арбуз банан вишня груша дыня ежевика женьшень земляника ирис картофель
   Вывести содержимое коллекции на экран, каждый элемент с новой строки.
   Посмотреть, как изменился порядок добавленных элементов.

2. Создать коллекцию **Map**<String, String> (реализация HashMap), занести туда **10** пар строк:
   арбуз - ягода, банан - трава, вишня - ягода, груша - фрукт, дыня - овощ, ежевика - куст, жень-шень - корень, земляника - ягода, ирис - цветок, картофель - клубень.
   Вывести содержимое коллекции на экран, каждый элемент с новой строки.
   Пример вывода (тут показана только одна строка):
   картофель - клубень

3. Есть коллекция **Map**<String, String> (реализация HashMap), туда занесли **10 различных строк**. Вывести на экран отдельно список ключей и отдельно список значений, каждый элемент с новой строки.

```
public class Solution {
   public static void main(String[] args) {
      Map<String, String> map = new HashMap<>();
      map.put("Simk", "Sim");
      map.put("Tomk", "Tom");
      map.put("Arbusk", "Arbus");
      map.put("Babyk", "Baby");
      map.put("Catk", "Cat");
      map.put("Dogk", "Dog");
      map.put("Eatk", "Eat");
      map.put("Foodk", "Food");
      map.put("Geveyk", "Gevey");
      map.put("Hugsk", "Hugs");
      printKeys(map);
      printValues(map);
}
public static void printKeys(Map<String, String> map) {
   //напишите тут ваш код

}
public static void printValues(Map<String, String> map) {
   //напишите тут ваш код

}
}
```

4. Для **arrayList** и **linkedList** провести **10** тысяч **вставок**, **удалений**, а также вызовов get и set.

```
public class Solution {
   public static void main(String[] args) {
      // ArrayList
      ArrayList arrayList = new ArrayList();
      insert10000(arrayList);
      get10000(arrayList);
      set10000(arrayList);
      remove10000(arrayList);
      // LinkedList
      LinkedList linkedList = new LinkedList();
      insert10000(linkedList);
      get10000(linkedList);
      set10000(linkedList);
      remove10000(linkedList);
   }

   public static void insert10000(List list) {
      //напишите тут ваш код
   }

   public static void get10000(List list) {
      //напишите тут ваш код
   }

   public static void set10000(List list) {
      //напишите тут ваш код
   }

   public static void remove10000(List list) {
      //напишите тут ваш код
   }
}
```

5. Измерить, сколько времени занимает **10** тысяч вставок для каждого списка. Метод getInsertTimeInMs должен вернуть время своего исполнения в миллисекундах.

```
public class Solution {
   public static void main(String[] args) {
      System.out.println(getInsertTimeInMs(new ArrayList()));
      System.out.println(getInsertTimeInMs(new LinkedList()));
   }

   public static long getInsertTimeInMs(List list) {
      // напишите тут ваш код
      insert10000(list);
      // напишите тут ваш код
   }

   public static void insert10000(List list) {
      for (int i = 0; i < 10000; i++) {
         list.add(0, new Object());
      }
   }
}
```

6. Реализовать **4 метода**. Каждый из них должен возвращать **список**, который лучше всего подходит для выполнения данных операций (быстрее всего справится с большим количеством операций).

```
public class Solution {
   public static List getListForGet() {
      //напишите тут ваш код
   }

   public static List getListForSet() {
      //напишите тут ваш код
   }

   public static List getListForAddOrInsert() {
      //напишите тут ваш код
   }

   public static List getListForRemove() {
      //напишите тут ваш код
   }

   public static void main(String[] args) {

   }
}
```

7. Измерить, сколько времени занимает **10** тысяч вызовов get для каждого списка. Метод getGetTimeInMs должен вернуть время своего исполнения в миллисекундах.

```
public class Solution {
   public static void main(String[] args) {
      System.out.println(getGetTimeInMs(fill(new ArrayList())));
      System.out.println(getGetTimeInMs(fill(new LinkedList())));
   }

   public static List fill(List list) {
      for (int i = 0; i < 10000; i++) {
         list.add(new Object());
      }
      return list;
   }

   public static long getGetTimeInMs(List list) {
      // напишите тут ваш код
      get10000(list);
      // напишите тут ваш код
   }

   public static void get10000(List list) {
      if (list.isEmpty()) {
         return;
      }
      int x = list.size() / 2;
      for (int i = 0; i < 10000; i++) {
         list.get(x);
      }
   }
}
```

8. 1. Создайте список чисел.
   2. Добавьте в список **10 чисел** с клавиатуры.
   3. Вывести на экран длину самой длинной последовательности повторяющихся чисел в списке.
   Пример для списка 2, 4, 4, 4, 8, 8, 4, 12, 12, 14 :
   Искомое значение равно **3**, т.к. самая длинная последовательность повторяющихся чисел состоит из трех четверок.
   Если не повторяющиеся, то длина = 1.

9. Создать словарь (**Map**<String, String>) занести в него десять записей по принципу "**Фамилия**" - "**Имя**". Проверить сколько людей имеют **совпадающие** с заданным именем или фамилией.

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

    public static int getCountTheSameFirstName(Map<String, String> map, String name) {
        //напишите тут ваш код
    }

    public static int getCountTheSameLastName(Map<String, String> map, String lastName) {
        //напишите тут ваш код
    }

    public static void main(String[] args) {

    }
}
```

10. Создать словарь (**Map**<String, Date>) и занести в него десять записей по принципу: "**фамилия**" - "**дата рождения»**. Удалить из словаря всех людей, родившихся летом.

```
public class Solution {
   public static Map<String, Date> createMap() throws ParseException {
      DateFormat dateFormat = new SimpleDateFormat("MMMMM d yyyy", Locale.ENGLISH);
      Map<String, Date> map = new HashMap<>();
      map.put("Сталлоне", dateFormat.parse("MAY 1 2012"));
      //напишите тут ваш код
   }

   public static void removeAllSummerPeople(Map<String, Date> map) {
      //напишите тут ваш код
   }

   public static void main(String[] args) {

   }
}
```

11. Создать словарь (**Map**<String, String>) занести в него десять записей по принципу "**фамилия**" - "**имя**". Удалить людей, имеющих одинаковые имена.

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

   public static void removeTheFirstNameDuplicates(Map<String, String> map) {
      //напишите тут ваш код
   }

   public static void removeItemFromMapByValue(Map<String, String> map, String value) {
      Map<String, String> copy = new HashMap<>(map);
      for (Map.Entry<String, String> pair : copy.entrySet()) {
         if (pair.getValue().equals(value)) {
            map.remove(pair.getKey());
         }
      }
   }

   public static void main(String[] args) {

   }
}
```

12. Создать словарь (**Map**<String, Integer>) и занести в него десять записей по принципу: "**фамилия**" - "**зарплата**". Удалить из словаря всех людей, у которых зарплата ниже **500**.

```
public class Solution {
   public static Map<String, Integer> createMap() {
      //напишите тут ваш код
   }

   public static void removeItemFromMap(Map<String, Integer> map) {
      //напишите тут ваш код
   }

   public static void main(String[] args) {

   }
}
```

13. 1. Создать словарь **Map** (<String, String>) и добавить туда **10** человек в виде "**Фамилия**"-"**Имя**".
    2. Пусть среди этих **10 человек** есть люди с одинаковыми именами.
    3. Пусть среди этих **10 человек** есть люди с одинаковыми фамилиями.
    4. Вывести содержимое **Map** на экран.

```
public class Solution {
   public static void main(String[] args) {
      Map<String, String> map = createPeopleMap();
      printPeopleMap(map);
   }

   public static Map<String, String> createPeopleMap() {
      //напишите тут ваш код
      return null;
   }

   public static void printPeopleMap(Map<String, String> map) {
      //напишите тут ваш код
   }
}
```

14. Переставьте один модификатор **static**, чтобы пример скомпилировался.

```
public class Solution {
   public int A = 5;
   public static int B = 2;

   public int C = A * B;
   public static int D = B * A;

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

   public static int getValue() {
      return D;
   }

   public int getValue2() {
      return C;
   }
}
```

15. Создать **массив** на **20 чисел**. Заполнить его **числами** с клавиатуры. Вывести пять наибольших чисел. Каждое значение с новой строки.

```
public class Solution {
   public static void main(String[] args) throws Exception {
      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      int[] array = new int[20];
      for (int i = 0; i < array.length; i++) {
         array[i] = Integer.parseInt(reader.readLine());
      }
      sort(array);
      System.out.println(array[0]);
      System.out.println(array[1]);
      System.out.println(array[2]);
      System.out.println(array[3]);
      System.out.println(array[4]);
   }

   public static void sort(int[] array) {
      //напишите тут ваш код
   }
}
```

16. Программа вводит с клавиатуры **имя месяца** и выводит его номер на экран в виде: "**May is the 5 month**». Используйте коллекции.

17. Введите с клавиатуры **20 слов** и выведите их в **алфавитном порядке**. Каждое слово - с новой строки.

18. 1. Ввести с клавиатуры число N.
    2. Считать N целых чисел и заполнить ими список - метод getIntegerList.
    3. Найти минимальное число среди элементов списка - метод getMinimum.

```
public class Solution {
   public static void main(String[] args) throws Exception {
      List<Integer> integerList = getIntegerList();
      System.out.println(getMinimum(integerList));
   }

   public static int getMinimum(List<Integer> array) {
      // Найти минимум тут
      return 0;
   }

   public static List<Integer> getIntegerList() throws IOException {
      // Создать и заполнить список тут
      return null;
   }
}
```

19. Задача: Программа определяет, какая **семья** (фамилию) живёт в доме с указанным номером.
Новая задача: Программа должна работать не с **номерами домов**, а с **городами**:
Пример ввода:
Москва
Ивановы
Киев
Петровы
Лондон
Абрамовичи

Лондон
Пример вывода:
Абрамовичи

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

        List<String> list = new ArrayList<>();
        while (true) {
            String family = reader.readLine();
            if (family.isEmpty()) {
                break;
            }
            list.add(family);
        }

        // Read the house number
        int houseNumber = Integer.parseInt(reader.readLine());

        if (0 <= houseNumber && houseNumber < list.size()) {
            String familyName = list.get(houseNumber);
            System.out.println(familyName);
        }
    }
}
```

20. 1. Реализовать метод isDateOdd(String **date**) так, чтобы он возвращал **true**, если количество дней с начала года - нечетное число, иначе **false**
2. String **date** передается в формате **FEBRUARY 1 2013**
Не забудьте учесть первый день года.
Пример:
JANUARY 1 2000 = true
JANUARY 2 2020 = false

```
public class Solution {
   public static void main(String[] args) {
      System.out.println(isDateOdd("MAY 1 2013"));
   }

   public static boolean isDateOdd(String date) {
      return true;
   }
}
```

21. 1. Создайте класс **Human** с полями **имя** (String), **пол** (boolean), **возраст** (int), **дети** (ArrayList<Human>).
2. Создайте **объекты** и заполните их так, чтобы получилось: **два дедушки**, **две бабушки**, **отец**, **мать**, **трое детей**.
3. Выведите все объекты Human на экран (**Подсказка:** используй метод toString() класса Human).

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

   public static class Human {
      //напишите тут ваш код

      public String toString() {
         String text = "";
         text += "Имя: " + this.name;
         text += ", пол: " + (this.sex ? "мужской" : "женский");
         text += ", возраст: " + this.age;
         int childCount = this.children.size();
         if (childCount > 0) {
            text += ", дети: " + this.children.get(0).name;
            for (int i = 1; i < childCount; i++) {
               Human child = this.children.get(i);
               text += ", " + child.name;
            }
         }
         return text;
      }
   }
}
```

22. Написать программу, которая вводит с клавиатуры **строку текста**.
Программа заменяет в тексте первые буквы всех слов на заглавные.
Вывести результат на экран.
Пример ввода: мама мыла раму.
Пример вывода: Мама Мыла Раму

23. 1. Внутри класса Solution создать public static класс кот - **Cat**.
2. Реализовать метод createCats, он должен создавать множество (**Set**) котов и добавлять в него **3** кота.
3. В методе main удалите одного кота из **Set** cats.
4. Реализовать метод printCats, он должен вывести на экран всех котов, которые остались во множестве.
Каждый кот с новой строки.

```
public class Solution {
   public static void main(String[] args) {
      Set<Cat> cats = createCats();

      //напишите тут ваш код. step 3 - пункт 3

      printCats(cats);
   }

   public static Set<Cat> createCats() {
      //напишите тут ваш код. step 2 - пункт 2
      return null;
   }

   public static void printCats(Set<Cat> cats) {
      // step 4 - пункт 4
   }

   // step 1 - пункт 1
}
```

24. 1. Внутри класса **Solution** создать public static классы **Cat**, **Dog** без конструктора или с конструктором без параметров.
    2. Реализовать метод createCats, который должен возвращать множество с **4** котами.
    3. Реализовать метод createDogs, который должен возвращать множество с **3** собаками.
    4. Реализовать метод join, который должен возвращать объединенное множество всех животных - всех котов и собак.
    5. Реализовать метод removeCats, который должен **удалять** из множества **pets** всех котов, которые есть в множестве **cats**.
    6. Реализовать метод printPets, который должен выводить на экран **всех животных**, которые в нем есть.
    Каждое животное с новой строки.

```
public class Solution {
  public static void main(String[] args) {
    Set<Cat> cats = createCats();
    Set<Dog> dogs = createDogs();

    Set<Object> pets = join(cats, dogs);
    printPets(pets);

    removeCats(pets, cats);
    printPets(pets);
  }

  public static Set<Cat> createCats() {
    Set<Cat> result = new HashSet<Cat>();
    //напишите тут ваш код

    return result;
  }

  public static Set<Dog> createDogs() {
    //напишите тут ваш код
    return null;
  }

  public static Set<Object> join(Set<Cat> cats, Set<Dog> dogs) {
    //напишите тут ваш код
    return null;
  }

  public static void removeCats(Set<Object> pets, Set<Cat> cats) {
    //напишите тут ваш код
  }

  public static void printPets(Set<Object> pets) {
    //напишите тут ваш код
  }

  //напишите тут ваш код
}
```