دستور کار شماره ۵

Immutable objects

- Define private as access class fields
- Make fields “final”. In Java, "final" means something is unchangeable or cannot be overridden. It can be used with variables to create constants or with methods to prevent them from being overridden.
- In the constructor of the class, we initialize all the fields so that there is no need for another method for initialization (so no need for setters)
- Implement getter methods for non-primitive fields in a way that instead of returning the object itself, it creates and returns a copy of the same object. However, for classes like String that are immutable, there is no need for this approach.

Collections in java

- Collections are used to manage and store objects
- Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.
- Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).
- We will define 4 types of collections in this workshop:
    - ArrayList
    - LinkedList
    - HashSet
    - HashMap

ArrayList

- In Java, ArrayList is a class that is a class that implements the List interface. It is a dynamic array-like data structure that allows you to store and manipulate elements in an ordered sequence
- to use ArrayList you need to first import it. from the utils library:
  - `import java.util.ArrayList`
- this is how we made array of strings:
  - `String[] strings = new String[size];`
- this is how we make an ArrayList of strings:
  - `ArrayList<String> strings = new ArrayList();`
- we can use the our own custom classes inside <>, for example:
  - `ArrayList<Person> persons = new ArrayList<>();`
- we cannot use primitive types inside <> for example we cannot declare:
  - `ArrayList<int> ints = new ArrayList(); // this is invalid`
  - instead use:
  - `ArrayList<Integer> ints = new ArrayList(); // this is valid`

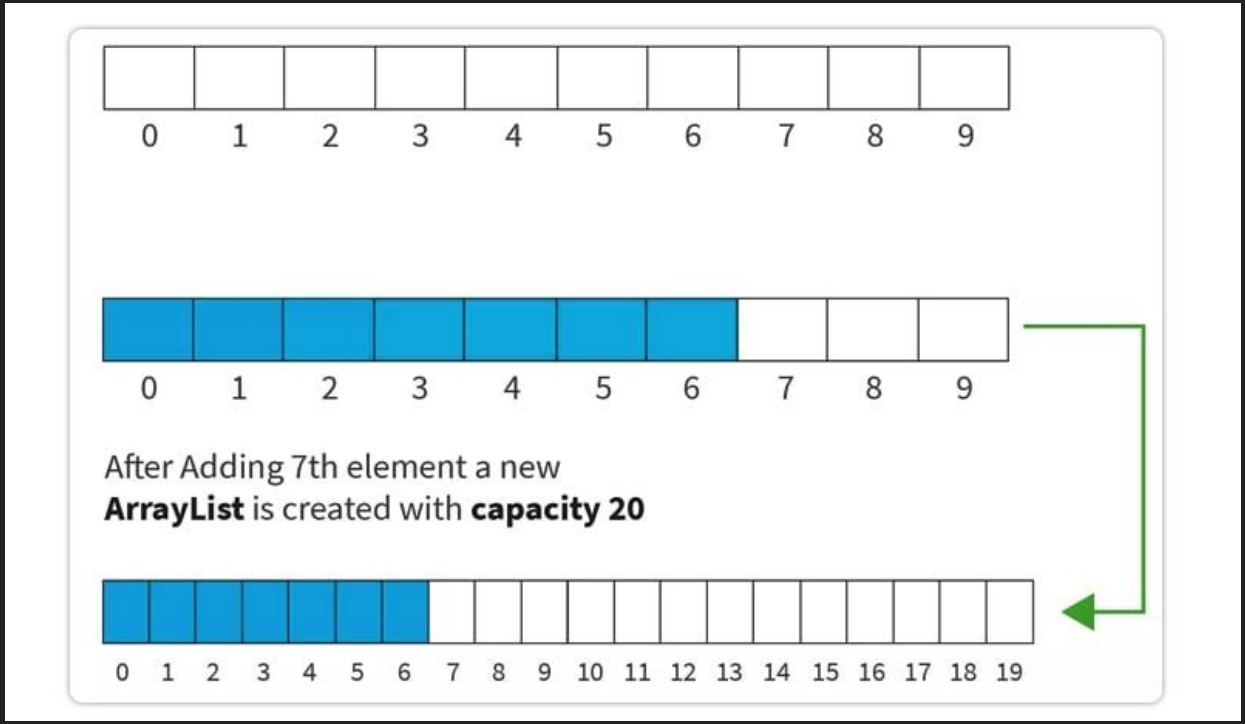
Example on how to use ArrayList

```
import java.util.ArrayList;
public class Company {
    private ArrayList<Employee> employees;
    public Company(){
        employees = new ArrayList<>();
    }

    public void addEmployee(Employee newEmployee){
        employees.add(newEmployee);
    }
    public void removeEmployee(int indexOfTheEmployee){
        employees.remove(indexOfTheEmployee);
    }
    public void removeEmployee(Employee employee){
        employees.remove(employee);
        // the employee is removed and the objects are shifted to the left
    }
    public void setEmployee(int index, Employee employee){
        employees.set(index, employee);
    }

    public int getNumberOfEmployees(){
        return employees.size();
    }
}
```

LinkedList

- the LinkedList class can be imported by using:
  - `import java.util.LinkedList`
- Most of the methods in LinkedList and ArrayList are the same
  - add()
  - remove()
  - set() and so on
- The difference is in the implementation of each

ArrayList architecture



ArrayList architecture

- When a Java ArrayList is created, its default capacity is 10 if not given or mentioned by the user. The capacity or the size of the ArrayList in Java increases based on the load factor and existing size.

  The Load Factor is a means to decide when to grow its size. The default value of a Java ArrayList's load factor is 0.75f

  A Java ArrayList increases its size after every entry, which is measured as the product of existing capacity and the ArrayList's load factor.

  Threshold = (Load Factor) * (Current Capacity)

  For instance, if a user creates a Java ArrayList of capacity 10

  Threshold = Load Factor * Current Capacity

  = 0.75 * 10

  ≅ 7

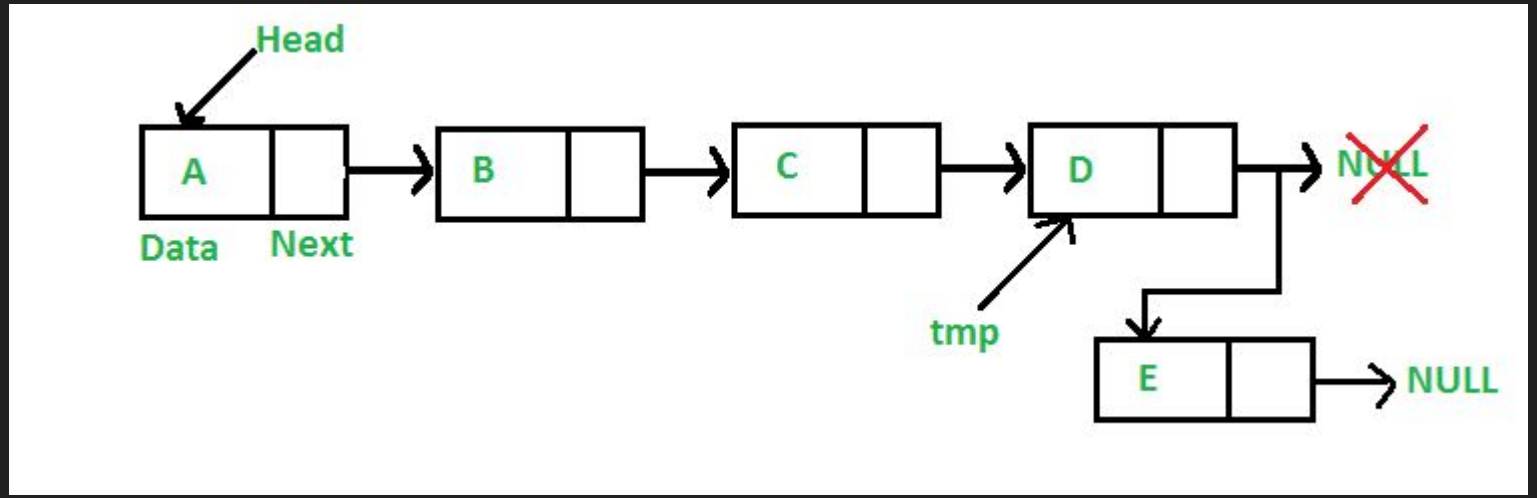
ArrayList architecture

- In Java 8 or in later versions, the new size of the ArrayList is determined to be 50% higher than its earlier capacity.

  new_capacity = old_capacity + (old_capacity >> 1)

- In the case of a Java ArrayList, manipulation is sometimes slower because removing any particular element from the list takes a lot of time.

LinkedList architecture



HashSet

- very similar to ArrayList with some key differences
- import using `import java.util.HashSet`
- ArrayList is implemented using a resizable array. It maintains the order of elements and allows duplicate elements. Elements are accessed using their index, providing fast indexed access.
- HashSet: HashSet does not provide direct access to elements through indexing. You can only iterate over the elements using iterators or enhanced for-each loop.

HashSet

- the elements in HashSet must be unique
- note that the order of the members is not preserved

```
cars.add(“Benz”);
cars.add(“Toyota”);
cars.add(“Ford”);
[“Benz”,”Ford”,”Toyota”]
```

- each object is unique

```
cars.add(“Toyota”);
[“Benz”,”Ford”,”Toyota”]
```

HashSet

iterate over the elements using foreach

```
for(String car: cars){
      System.out.println(car);
}
```

can be also used on ArrayLists

HashMap

import with the following command

```
import java.util.HashMap
```

HashMap provides a data structure for storing key-value pairs, allowing retrieval and modification of values based on their corresponding keys.

inside <> we put 2 class names.

HashMap

in the <key, value> pair, the keys must be different but the values can be the same.

```
HashMap<String,String> capitalCities = new HashMap<>();
// England is the key and London is the value
capitalCities.put("England", "London");
capitalCities.put("Germany", "Berlin");
// getting the value using the key
capitalCities.get("Germany");
// removing the key value pair from capitalCities
capitalCities.remove("Germany);
```

HashMap

iterating on keys:

```
for(String country : capitalCities.keySet()){
print(capitalCities.get(country);
}
```

iterating on values:

```
for(String country : capitalCities.values()){
print(capitalCities.get(country);
}
```

Iterator

Suppose we want to iterate an arraylist of strings and remove the ones that contain the word “java”. you might be thinking of something like this:

```
for(String s :strings){
      if(s.contains("java"){
            strings.remove(s);
      }
}
```

but this code will not work, the foreach cannot handle updating index while the arraylist is being shifted. to solve this problem we use iterators

گزارش خود را در قالب زیر ارسال کنید.

ایمیل: amir.frg.77@gmail.com

تلگرام: @amirf80

- Workshop5_StdNo_StdName.zip

در گزارش کار خود اسکرین شات ورودی و خروجی برنامه را قرار دهید.

```
Workshop5_StdNo_StdName
├── Codes
└── report.pdf
```

- ایجاد یک VotingSystem
- ایجاد یک Voting به صورت ناشناس و یک Voting به صورت عادی
- رای دهی توسط دو نفر در هر کدام از Voting ها
- برای گرفتن تاریخ فعلی می‌توانید از کلاس Date استفاده کنید:

```
import java.util.Date;

Date date = new Date();

String dateAsString = date.toString();
```