

پرناهه سازی پیش روی



ظرفها

مهدی مصطفیزاده

سرفصل مطالب

- رده‌های ArrayList, LinkedList, HashSet, HashMap
- واسطه‌های Collection, List, Set, Map
- مفهوم hashCode و متدهای Hashing
- پیمایشگر (Iterator)
- کاربرد Comparable و Comparator در مقایسه اشیاء
- رده‌های Arrays و Collections



چارچوب مجموعه‌های جاوا

چارچوب مجموعه‌های جاوا (Java Collections Framework)

مجموعه‌ای از رده‌ها و واسطه‌ای برای نگهداری مجموعه‌هایی از اشیاء و مدیریت آنها

مدیریت: پیمایش، افروختن، حذف، جستجو، ویرایش و ...

به رده‌های نگهدارنده مجموعه‌ای از اشیاء، طرف (Container) می‌گویند.



طرح مساله

- می‌خواهیم فهرستی از اشیاء، را در یک ساختمان داده ذخیره کنیم و عملیات لازم را پر روی آن انجام دهیم.
- اشیاء می‌توانند تکراری باشند.
- ترتیب اشیاء اهمیت دارد.
- راهکار اول: استفاده از آرایه
- مشکل: تپت پودن طول آرایه



ArrayList: راه حل:

- ردای که امکان نگهداری فهرست مرتبا از اشیاء (در واقع ارجاعات په اشیاء) را دارد.
- امکان نگهداری اشیاء تکراری (اشیائی که پر اساس متد equal با هم پر ابرند) را دارد.
- خود ArrayList برای نگهداری اشیاء از آرایه استفاده می کند.
- طول پیشفرض آرایه 10 (در زمان ایجاد توسط ورودی سازنده قابل تعییر است)



```
ArrayList<String> list = new ArrayList<>();
System.out.println(list.size()); // 0
list.add("ali");
list.add("ali");
list.add("ahmad");
System.out.println(list.size()); // 3
list.add(1, "hassan");
System.out.println(list.size()); // 4
System.out.println(list.get(1)); // hassan
System.out.println(list.contains("hassan")); // true
System.out.println(list.indexOf("ali")); // 0
System.out.println(list.lastIndexOf("ali"));
System.out.println(list.isEmpty()); // false
System.out.println(list.remove("ali")); // true
System.out.println(list.get(0)); // hassan
System.out.println(list.get(1)); // ali
System.out.println(list.size()); // 3
System.out.println(list.set(2, "akbar")); // ahmad
System.out.println(list.get(2)); // akbar
System.out.println(list.size()); // 3
System.out.println(list.remove(2)); // akbar
System.out.println(list.size()); // 2
```



ArrayList در مقابل LinkedList

- از پیروں و از منظر واسطه‌ها (مُنْدَهَا)، LinkedList هیچ تفاوتی پا ندارد.
- تفاوت پر سر پیاده‌سازی درونی است. LinkedList په جای آرایه از لیست پیوندی استفاده می‌کند.
- کارایی LinkedList در درج کردن‌های متعدد پهتدر از ArrayList است.
- کارایی ArrayList در دستیابی‌های متعدد په اشیاء، از طریق موقعیت (index)، پهتدر از LinkedList است.



واسط List

- با توجه به یکسان پودن متدهای `ArrayList` و `LinkedList`، یک واسط پالادستی تخته عنوان `List` تعریف شده است.
- هر دو ردهی `ArrayList` و `LinkedList` را پیادهسازی کرده‌اند.
- متدهایی که پا هم دیدیم، در واسط `List` تعریف شده‌اند.
- استفاده از `List` به جای استفادهی مستقیم از `ArrayList` یا `LinkedList` سبب تحقق DIP می‌شود.



طرح مساله

- می خواهیم مجموعه ای از اشیاء را در یک ساختمان داده ذخیره کنیم و عملیات لازم را پر روی آن انجام دهیم.
- اشیاء نمی توانند تکراری باشند.
- ترتیب اشیاء اهمیت ندارد.
- راهکار اول: استفاده از ردیف HashSet



Set و HashSet

- برای نگهداری مجموعه‌ای از اشیاء، غیرتکراری استفاده می‌شود.
- ترتیب در آن اهمیتی ندارد؛ بنابراین `index` در آن موضوعیت ندارد.
- متد های آن مشابه متد های `ArrayList` است؛ با این تفاوت که متد های وابسته به `index` در آن تعریف نشده‌اند.
- متد های مذکور در حقیقت در واسطه `Set` تعریف شده‌اند و `HashSet` آنها را پیاده‌سازی کرده است.



(مثال) HashSet

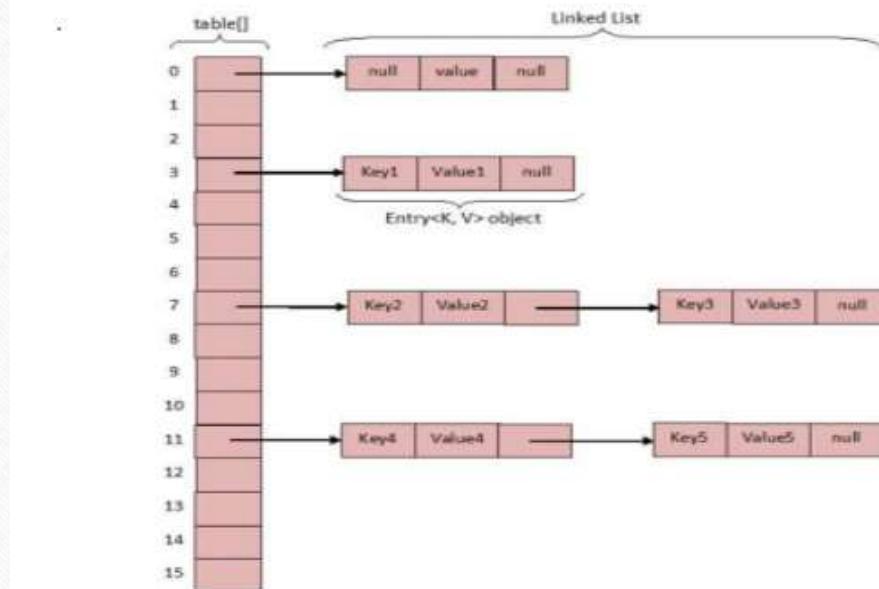
```
7 public static void main(String[] args) {  
8     Set<Person> people = new HashSet<>();  
9     people.add(new Person("ali"));  
10    people.add(new Person("ali"));  
11    System.out.println(people.size());  
12    System.out.println(people.contains(new Person("ali")));  
  
<terminated> Main [Java Application] F:\Mahdi\Java\jdk1.8.0_112\bin\javaw.exe (Jun 1, 2020, 12:04:31 PM)  
2  
false
```

چطور شد؟ *

```
public class Person {  
    private String name;  
  
    public Person(String name) {  
        this.name = name;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    @Override  
    public boolean equals(Object obj) {  
        if (!(obj instanceof Person))  
            return false;  
        return name.equals(((Person)obj).name);  
    }  
}
```



سازوکار HashMap و HashSet در Hashing



(تصحیح مثال) HashSet

```
7 public static void main(String[] args) {  
8     Set<Person> people = new HashSet<>();  
9     people.add(new Person("ali"));  
10    people.add(new Person("ali"));  
11    System.out.println(people.size());  
12    System.out.println(people.contains(new Person("ali")));
```

```
Mar... Pro... Serv... Dat... Sni... Pro... Con... Sea... JUnit Deb... Hist...  
<terminated> Main [Java Application] F:\Mahdi\Java\jdk1.8.0_112\bin\javaw.exe (Jun 1, 2020, 12:16:50 PM)  
1  
true
```

hashCode متد •

```
@Override  
public boolean equals(Object obj) {  
    if(!(obj instanceof Person))  
        return false;  
    return name.equals(((Person)obj).name);  
}  
  
@Override  
public int hashCode() {  
    return name.hashCode();  
}
```



Map و HashMap

- رده‌ای است که مجموعه‌ای از زوج‌های کلید و مقدار را در خود چای می‌دهد.
- سازوکار ذخیره و پارسیابی آن مبتنی بر Hashing روی کلیدهاست و از این رو، سرعت دستیابی به مقدار متناظر یک کلید، بالاست.
- متدات `Map` در حقیقت در واسط `HashMap` تعریف شده‌اند و آن‌ها را پیاده‌سازی کرده است.



```
Map<Person, City> people = new HashMap<>();
people.put(new Person("ali"), new City("Tehran"));
people.put(new Person("ahmad"), new City("Mashhad"));
people.put(new Person("ali"), new City("Mashhad"));
System.out.println(people.size()); // 2
System.out.println(people.get(new Person("ali"))); // Mashhad
System.out.println(people.containsKey(new Person("ahmad"))); // true
System.out.println(people.containsValue(new City("Mashhad"))); // true
for(Person p : people.keySet()) {
    System.out.print(p + " ");
} // ahmad ali
for(City c : people.values()) {
    System.out.print(c + " ");
} // Mahhad Mashhad
System.out.println();
System.out.println(people.remove(new Person("ahmad"))); // Mashhad
```



مقایسه‌ی اشیاء

- گاهی اوقات، پرخی از متدها از رده‌های مختلف نیازمند پارامتری ترتیب‌پذیر هستند؛ یعنی پارامتری از جنس رده‌ای که اشیا، آن قابلیت مقایسه دارند.
- در این موضع، پارامتر منظور از جنس واسط Comparable تعریف می‌شود.
- در این واسط، متدهی با عنوان compareTo تعریف شده است که امکان مقایسه‌ی شیء چاری با پارامتر ورودی را فراهم می‌آورد.
- خود چیز متبت به معنای بزرگتر پومن شیء چاری، خود چیز منفی به معنای کوچکتر پومن شیء چاری و خود چیز صفر به معنای تساوی).



مقایسه اشیاء (مثال)

```
8* public static void main(String[] args) {
9     List<Person> people = new ArrayList<>();
10    people.add(new Person("ali", 20));
11    people.add(new Person("hassan", 12));
12    people.add(new Person("ahmad", 18));
13    System.out.println(people);
14    Collections.sort(people);
15    System.out.println(people);
```

```
public class Person implements Comparable<Person> {
    private String name;
    private int age;

    @Override
    public int compareTo(Person p) {
        return age - p.age;
    }

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

terminated> Main [Java Application] F:\Mahdi\Java\jdk1.8.0_112\bin\javaw.exe (Jun 1, 2020, 1:40:21)
[ali, hassan, ahmad]
[hassan, ahmad, ali]



مقایسه‌ی اشیاء با Comparator

- گاهی اوقات، یک متند نیاز به دانستن ترتیب اشیاء ورودی دارد؛ اما رده‌ی اشیاء ورودی Comparable نیست.
- ممکن است رده‌ی اشیاء Comparable باشد، اما منطق compareTo در آن نقطه از کد، مطلوب نباشد.
- به عنوان نمونه، در مثال قبل، ممکن است بعوهایم مرتب‌سازی براساس طول قدر انجام شود (و نه سر).
- در چنین موقعی می‌توان از واسط Comparator استفاده کرد.



مقایسه اشیاء با Comparator (مثال)

```
public static void main(String[] args) {
    List<Person> people = new ArrayList<>();
    people.add(new Person("ali", 20, 190));
    people.add(new Person("hassan", 12, 180));
    people.add(new Person("ahmad", 18, 170));
    System.out.println(people);
    Collections.sort(people);
    System.out.println(people);
    Collections.sort(people, new Comparator<Person>() {
        @Override
        public int compare(Person p1, Person p2) {
            return p1.getHeight() - p2.getHeight();
        }
    });
    System.out.println(people);
}
```

```
ic class Person implements Comparable<Person> {
    private String name;
    private int age;
    private int height;

    @Override
    public int compareTo(Person p) {
        return age - p.age;
    }

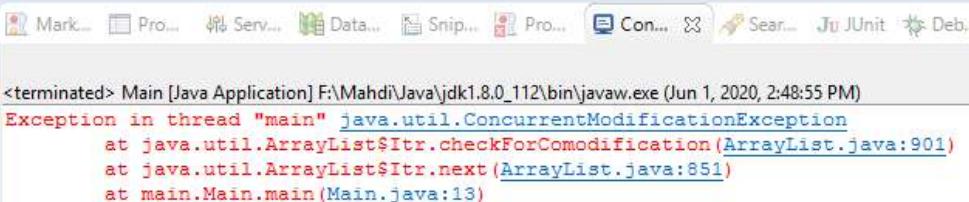
    public Person(String name, int age, int height) {
        this.name = name;
        this.age = age;
        this.height = height;
    }
}
```

[ali, hassan, ahmad]
[hassan, ahmad, ali]
[ahmad, hassan, ali]



طرح مساله

```
6 public class Main {  
7     public static void main(String[] args) {  
8         List<Person> people = new ArrayList<>();  
9         people.add(new Person("ali", 20, 190));  
10        people.add(new Person("ahmad", 18, 170));  
11        people.add(new Person("hassan", 12, 180));  
12  
13        for(Person p : people) {  
14            if(p.getAge() < 15)  
15                people.remove(p);  
16        }  
17    }  
18}
```



The screenshot shows a Java application running in an IDE. The code removes all persons under 15 years old from a list. However, it fails at line 13 due to a ConcurrentModificationException.

```
<terminated> Main [Java Application] F:\Mahdi\Java\jdk1.8.0_112\bin\javaw.exe (Jun 1, 2020, 2:48:55 PM)  
Exception in thread "main" java.util.ConcurrentModificationException  
at java.util.ArrayList$Itr.checkForComodification(ArrayList.java:901)  
at java.util.ArrayList$Itr.next(ArrayList.java:851)  
at main.Main.main(Main.java:13)
```

- می خواهیم لیست افراد را پیش کرده و افرادی که سن آنها کمتر از ۱۵ سال است، از لیست حذف کنیم.

ConcurrentModificationException

چه؟



پیمایش ظروف با Iterator

```
10  List<Person> people = new ArrayList<>();
11  people.add(new Person("ali", 20, 190));
12  people.add(new Person("ahmad", 18, 170));
13  people.add(new Person("hassan", 12, 180));
14
15  Iterator<Person> iter = people.iterator();
16  while(iter.hasNext()) {
17      Person currentPerson = iter.next();
18      if(currentPerson.getAge() < 15)
19          iter.remove();
20  }
21  System.out.println(people);
22
```

The screenshot shows a Java application running in an IDE. The code uses an Iterator to traverse a list of Person objects. It removes any person who is 15 years or younger. The output window shows the remaining names: [ali, ahmad].

```
<terminated> Main [Java Application] F:\Mahdi\Java\jdk1.8.0_112\bin\javaw.exe (Jun 1, 2020, 3:48:46 PM)
[ali, ahmad]
```



ردیهای کمکی Collections و Arrays

```
List<Person> people = new ArrayList<>();
people.add(new Person("ali", 20, 190));
people.add(new Person("ahmad", 18, 170));
people.add(new Person("hassan", 12, 180));

Collections.sort(people);
System.out.println(
    Collections.binarySearch(people, new Person("ali", 20, 190))
); // 2

List<Person> unmodifiableList = Collections.unmodifiableList(people);
// UnsupportedOperationException:
// unmodifiableList.add(new Person("reza", 20, 175));

Integer[] numbers = {5, 10, 7};
Integer[] temp = new Integer[3];
List<Integer> numbersList = Arrays.asList(numbers);
Integer[] numbers2 = (Integer[]) numbersList.toArray();
Integer[] numbers3 = numbersList.toArray(temp);
System.out.println(temp[0]); // 5
System.out.println(numbers3[0]); // 5
```

• مجموعه‌ای از متدهای ایستا

