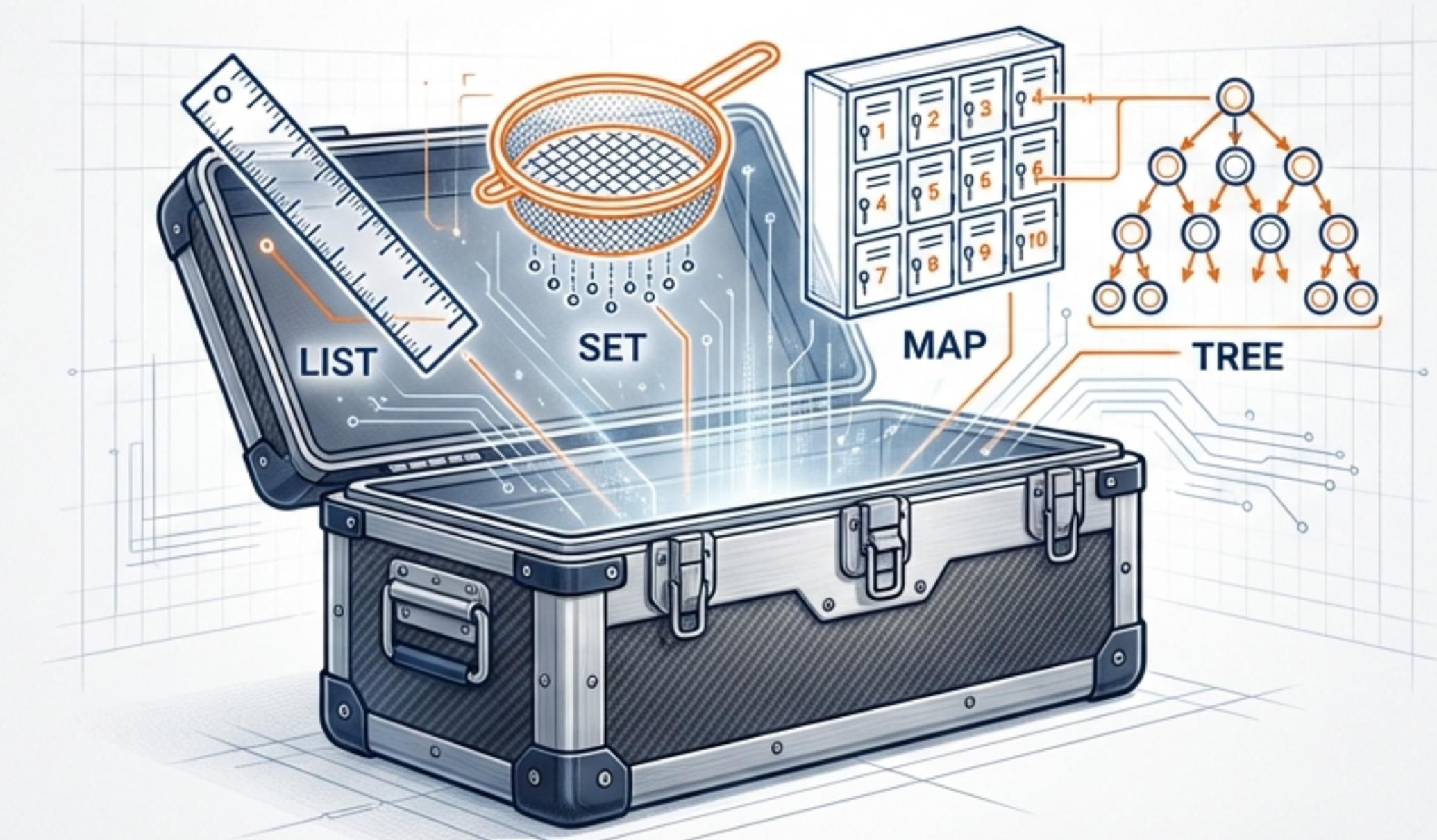


بخش ۱۶: چارچوب کلکسیون‌ها (Collections Framework)

– جعبه‌ابزار داده‌های پویا

تهییه شده توسط: سید سجاد پیراھش



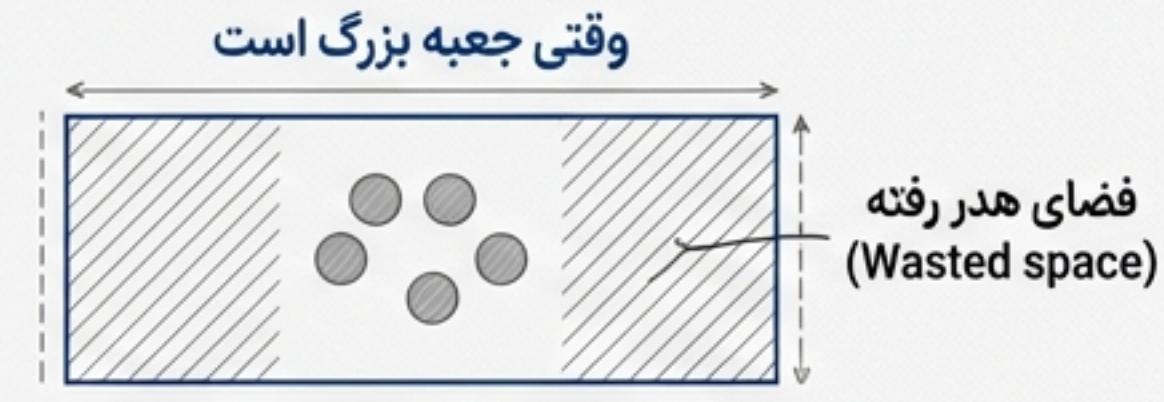
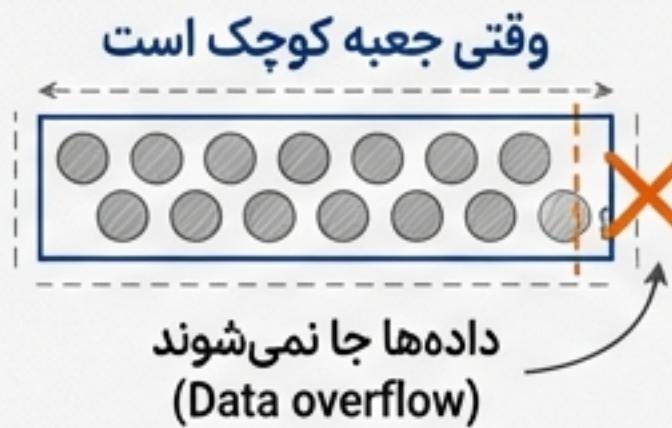
ابزار قدیمی ما: محدودیت فلجهنده آرایه‌ها



آرایه (Array)

آرایه‌ها قدرتمند و سریع هستند، اما یک محدودیت بزرگ دارند: اندازه ثابت (Fixed Size).

در دنیای واقعی، ما به ندرت اندازه دقیق داده‌ها را از قبل می‌دانیم.



شما در **همان لحظه** ساخت یک آرایه، باید **اندازه آن را مشخص کنید** و این اندازه **هرگز قابل تغییر نیست**. <<

جدول مقایسه Collection و Array

معیار	Collection (کلکسیون)	آرایه (Array)
اندازه (Size)	ثابت: در زمان ساخت تعیین می‌شود و قابل تغییر نیست.	پویا (Dynamic): می‌تواند بر اساس نیاز، رشد یا کوچک شود.
نوع داده	می‌تواند هم انواع اولیه (int) و هم اشیاء (String) را لگه دارد.	فقط اشیاء: نمی‌تواند انواع اولیه را مستقیماً نگه دارد (باید از Wrapper classes مثل Integer استفاده کرد).
ذخیره‌سازی	حافظه پیوسته (Contiguous).	ساختارهای مختلف (آرایه، لیست پیوندی، جدول هش).
متدها	تقریباً هیچ متدهی ندارد (فقط فیلد length).	مجموعه‌ای غنی از متدها برای افزودن، حذف، جستجو و ...
کارایی	برای دسترسی مستقیم با اندیس بسیار سریع است.	کارایی بستگی به نوع پیاده‌سازی دارد (مثلًا ArrayList در مقابل LinkedList).
الگوریتم‌ها	نیاز به پیاده‌سازی دستی الگوریتم‌ها (مرتب‌سازی، جستجو).	کلاس Collections ابزارهای آماده برای الگوریتم‌های رایج فراهم می‌کند.
Generics	پشتیبانی می‌کند، اما در ذات خود Type-safe نیست.	به شدت بر Generics برای تامین امنیت نوع (Type Safety) تکیه دارد.
سلسله‌مراتب	یک شیء ساده در جاوا است.	یک چارچوب کامل با رابطه‌ها و کلاس‌های تو در تو است.

ارتقاء به سطح حرفه‌ای: معرفی چارچوب کلکسیون‌ها

The Java Collections Framework is your new toolbox—a powerful, flexible, and standard set of tools for any data-handling job.



پویا (Dynamic): می‌تواند بر اساس نیاز، رشد یا کوچک شود.

قدرتمند (Powerful): مجموعه‌ای غنی از متدها و الگوریتم‌های آماده برای افزودن، حذف و جستجو.

استاندارد (Standard): ابزارهای تست‌شده و بهینه‌شده در خود جاوا، برای حل تقریباً هر مشکلی در زمینه ذخیره‌سازی داده‌ها.

اول ایمنی، بعد کار: Generics، نگهبان نوع داده شما

Generics (<>) are your essential safety gear. They provide **Type Safety** (امنیت نوع) by ensuring you only put the correct type of data into a collection, preventing `ClassCastException` at runtime.

کد خطرناک (قبل از Generics)

```
List names = new ArrayList();
names.add("سارا");
names.add(Integer.valueOf(123));
X خطری که دیده نمی‌شود! // این خط در زمان اجرا با خطا مواجه می‌شود
String name = (String) names.get(1); // ClassCastException!
```

کد امن و تمیز (با Generics)

```
List<String> names = new ArrayList<>();
names.add("سارا");
\ کامپایلر اجازه اضافه کردن نوع اشتباه را نمی‌دهد
// names.add(Integer.valueOf(123)); // ERROR
String name = names.get(0); // کاملاً امن
```

قبل از Generics، کلکسیون‌ها هر نوع شیئی را قبول می‌کردند و هنگام بازیابی داده، نیاز به Cast داشتند. این مشکل را در زمان کامپایل حل می‌کنند.

نقشه جعبه ابزار: آشنایی با سه محفظه اصلی

Your toolbox is organized into three main compartments: 'List', 'Set', and 'Map'. Understanding their core purpose is the key to choosing the right tool.



مقایسه مشخصات سه محفظه اصلی

معیار	List<E> (لیست)	Set<E> (مجموعه)	Map<K, V> (نگاشت)
مفهوم اصلی	یک دنباله مرتب از عناصر.	یک مجموعه از عناصر یکتا.	مجموعه‌ای از جفت‌های کلید-مقدار.
عناصر تکراری	✓ مجاز است.	✗ مجاز نیست.	کلیدها باید یکتا باشند، مقادیر می‌توانند تکراری باشند.
ترتیب	✓ حفظ می‌شود (ترتیب درج).	معمولًا حفظ نمی‌شود.	معمولًا حفظ نمی‌شود.
دسترسی	`get(int index)`	دسترسی مستقیم وجود ندارد (باید پیمایش شود).	با استفاده از کلید `get(Object key)`
پیاده‌سازی‌های رایج	`ArrayList`, `LinkedList`	`HashSet`, `TreeSet`	`HashMap`, `TreeMap`
ساختار null	می‌تواند چندین عنصر null داشته باشد.	حداکثر یک عنصر null می‌تواند داشته باشد.	یک کلید null و چندین مقدار null مجاز است.
کاربرد	زمانی که ترتیب عناصر مهم است و به دسترسی با انديس نياز دارد.	زمانی که می‌خواهيد از وجود عناصر تکراری جلوگيری کنيد.	زمانی که می‌خواهيد داده‌ها را بر اساس یک شناسه يكتا (کلید) ذخیره و بازیابی کنيد.

محفظه اول: – ابزاری برای داده‌های مرتب

Use a `List` when the order of elements matters and duplicates are allowed. The two main tools are `ArrayList` for fast access and `LinkedList` for fast insertions/deletions at the ends.



برگه مشخصات ابزار: LinkedList در مقابل ArrayList

معیار	ArrayList<E>	LinkedList<E>
ساختار داخلی	بر پایه یک آرایه پویا کار می‌کند.	بر پایه یک لیست پیوندی دوطرفه (گره‌ها) کار می‌کند.
دسترسی (get)	بسیار سریع ($O(1)$) چون مستقیماً به اندیس آرایه دسترسی دارد.	کند ($O(n)$) چون باید از ابتدا یا انتهای گره‌ها را یکی یکی طی کند.
درج/حذف (ابتدا/انتها)	کند ($O(n)$) اگر در ابتدا باشد، چون نیاز به شیفت دادن تمام عناصر دارد.	بسیار سریع ($O(1)$) چون فقط نیاز به تغییر چند اشاره‌گر دارد.
صرف حافظه	کمتر: فقط حافظه خود آرایه را مصرف می‌کند.	بیشتر: علاوه بر داده، برای هر گره حافظه اضافه برای اشاره‌گرهای prev و next مصرف می‌کند.
چه زمانی استفاده شود؟	اکثر موارد. به خصوص زمانی که عملیات اصلی شما خواندن و پیمایش لیست است.	زمانی که عملیات اصلی شما درج و حذف مکرر از ابتدا یا انتهای لیست است.

ابزار همه‌کاره: **ArrayList**، انتخاب اول در ۹۰٪ موارد

ArrayList is your default, go-to tool for most list-based tasks due to its incredibly fast random access speed ($O(1)$).

```
// ArrayList
List<String> studentNames = new ArrayList<>();

studentNames.add("آیدا");
studentNames.add("بابک");
studentNames.add("آیدا"); // تکرار مجاز است

// دسترسی برقآسا با اندیس
System.out.println("دانشجوی دوم" +
studentNames.get(1)); // -> بابک
```



نکته حرفه‌ای

در صنعت، اکثر عملیات روی لیست‌ها، خواندن و پیمایش خواندن و پیمایش (**get**) است، نه درج و حذف مکرر در ابتدا. به همین دلیل سرعت دسترسی آن را به **ArrayList** اول تبدیل کرده است.

محفظه دوم: — ابزاری برای تضمین یکتایی Set

Use a 'Set' when you absolutely cannot have duplicate elements. Order is generally not guaranteed.



عناصر تکراری؟ هرگز!

```
// HashSet از ایمیل‌های تکراری با  
Set<String> uniqueEmails = new HashSet<>();  
  
uniqueEmails.add("user@example.com");  
uniqueEmails.add("admin@example.com");  
uniqueEmails.add("user@example.com"); // گرفته می‌شود به صورت خودکار  
  
System.out.println(uniqueEmails.size()); // -> 2  
System.out.println(uniqueEmails); // ->  
-> [admin@example.com, user@example.com]
```

محفظه سوم: Map – ابزاری برای جستجوی فوری

Use a `Map` when you need to store and retrieve data using a unique key, like a dictionary or a phonebook.



جفت‌های کلید-مقدار (Key-Value Pairs)

کلیدها باید یکتا باشند.

مثال: دفترچه تلفن با
Map<String, String> phonebook = new
HashMap<>();

```
phonebook.put("مریم", "09121112233");  
phonebook.put("سینا", "09154445566");
```

جستجوی فوری با استفاده از کلید //
String sinasNumber = phonebook.get("سینا");

```
System.out.println("شماره سینا": " +  
sinasNumber); // -> "09154445566"
```

نگاهی به سرعت ابزارها: چرا جستجو در Set از List سریع‌تر است؟

The internal structure of a collection determines its performance. Hash-based collections ('HashSet', 'HashMap') are incredibly fast for searching because they don't have to check every element.



Linear Search: Requires checking each element until a match is found. Time, chatayes linearly ly with data elize.



Hash-based Search: Uses a hash function to uslatute the exact location tarulta. They is nearly instantaneous, instantanossi if defence, utarean of data size.

یک قانون مهم: کلکسیون‌ها فقط با اشیاء کار می‌کنند

You cannot use primitive types ('int', 'double', 'boolean') directly in collections. You must use their corresponding "Wrapper" class equivalents ('Integer', 'Double', 'Boolean').

کلاس پوششی (Wrapper)	نوع اولیه (Primitive)
int	Integer
double	Double
char	Character
boolean	Boolean

✗ اشتباه: از نوع اولیه نمی‌توان استفاده کرد

```
// اشتباه: از نوع اولیه نمی‌توان استفاده کرد  
List<int> numbers = new ArrayList<>();
```

✓ صحیح: از کلاس پوششی (Wrapper) استفاده کنید

```
// صحیح: از کلاس پوششی (Wrapper) استفاده کنید  
List<Integer> numbers = new ArrayList<>();
```

جاوا این تبدیل را به صورت خودکار برای شما انجام می‌دهد (Autoboxing)، اما شما باید در تعریف کلکسیون همیشه از نوع شیء (کلاس پوششی) استفاده کنید.

مهارت خود را بسنجید: کدام ابزار برای کدام کار؟

Let's apply what you've learned. For each scenario, choose the best collection type.



سناریو ۱: شما لیستی از مراحل یک دستور پخت را ذخیره می‌کنید. ترتیب اجرا مهم است و ممکن است یک مرحله تکرار شود.

- [ArrayList]
- [HashSet]
- [HashMap]



سناریو ۲: شما شناسه‌های کاربری (User ID) را برای یک سیستم ذخیره می‌کنید. هیچ دو کاربری نباید شناسه یکسان داشته باشند و سرعت چک کردن تکراری بودن شناسه حیاتی است.

- [ArrayList]
- [HashSet]
- [HashMap]



سناریو ۳: شما کدهای پستی را به نام شهرها نگاشت می‌کنید تا بتوانید با دادن کد پستی، نام شهر را به سرعت پیدا کنید.

- [ArrayList]
- [HashSet]
- [HashMap]

چالش استادی: پیاده‌سازی پشته (Stack) با LinkedList

Combine your knowledge. Use one tool from the toolbox (LinkedList) to build another common data structure (Stack).

The Challenge:

- با استفاده از یک `LinkedList<E>` داخلی، یک کلاس `MyStack<E>` بسازید.
- متدهای زیر را پیاده‌سازی کنید:

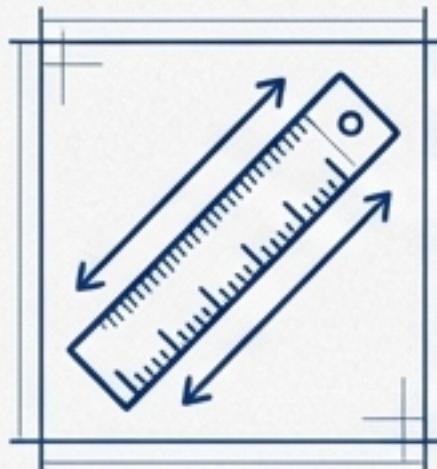
```
public class MyStack<E> {  
    private LinkedList<E> list = new LinkedList<E>();  
  
    public void push(E item) {  
        // ... your code here ...  
    }  
  
    public E pop() {  
        // ... your code here ...  
    }  
  
    public E peek() {  
        // ... your code here ...  
    }  
}
```

راهنمایی

از متدهای `(getLast(), removeLast(), addLast())` استفاده کنید.

خلاصه جعبه ابزار: راهنمای تقلب شما

Your quick-reference guide to the Java Collections Framework.



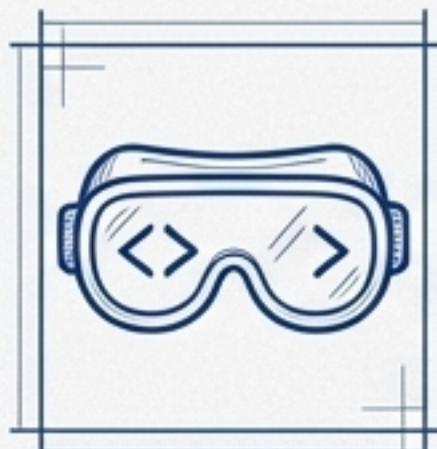
- مرتب، تکرار مجاز.
"List"
ArrayList (پیش فرض برای دسترسی سریع)،
LinkedList (برای درج/حذف سریع در ابتدا/انتها).



-> "Set"
یکتا، بدون ترتیب.
HashSet برای جلوگیری از تکرار و بررسی سریع عضویت.



->
"Map"
کلید-مقدار یکتا.
HashMap برای جستجوی فوری بر اساس کلید.



"Generics <>" ->
همیشه برای امنیت
(Type Safety) نوع استفاده کنید.



"Wrapper Classes" ->
برای ذخیره انواع اولیه از (**int**, **double**) ... و **Integer**, **Double** استفاده کنید.



".equals() ->
برای مقایسه محتوای اشیاء از **equals()** استفاده کنید، نه از **==**.

در جعبه ابزار بعدی: مدیریت شرایط غیرمنتظره

You've mastered dynamic data structures. Next, you'll learn how to build robust applications that can handle errors gracefully.

بخش بعدی: مدیریت استثناهای (Exception Handling)

