

# Java interview preparation

C:\Users\User-PC\IdeaProjects\untitled1

## Classes vs Instances

Classes are blueprints

Class dog

Attributes :String breed,Int age,String color

Methods:Bark()

**Watching: Class vs. instance**

2,1344,036

From the course: Nail Your Java Interview

1

**Dog A**  
Breed: "German Shepherd"  
Age: 3  
Color: Brown

2

**Dog B**  
Breed: "Golden Retriever"  
Age: 5  
Color: Yellow

LinkedIn

## Dog A and Dog B: Instances of Dog

- Dog A and Dog B both have a breed, age, and color, but the values of these different attributes are different
- Each dog can call the bark method
- Dog A and Dog B are instances of the Dog class

LinkedIn

Understanding the difference between  
a **class** and an **instance** is key.

LinkedIn

Static and Non Static methods

# Static vs. Nonstatic Methods

## Static method

- Class method
- Belongs to the class itself
- Do not need an instance in order to use a static method
- Method depends on class

## Nonstatic method

- Instance method
- Belongs to each object generated from the class
- Need an instance to use
- Method depends on individual characteristics of the object



# When to Use Nonstatic Methods

- Nonstatic methods can access nonstatic and static variables
- If your method is related to an object's attributes, it should be nonstatic



# When to Use Static Methods

- If you want to use a method without creating an instance of the class, static methods are the way to go
- Static methods can only access class variables



## Example

```
1 import java.util.Random;
2
3 public class Dice {
4     static int sidesOfDice = 6;
5     int faceValue = 0;
6
7     public int roll() {
8         Random rand = new Random();
9         this.faceValue = rand.nextInt(sidesOfDice) + 1;
10        return this.faceValue;
11    }
12
13    public static void changeNumSidesOfDice(int newNumberOfSides) {
14        Dice.sidesOfDice = newNumberOfSides;
15    }
16
17    public int getFaceValue() {
18        return this.faceValue;
19    }
20
21 }
```

The screenshot shows an IDE window titled 'HelloWorld' with a file explorer on the left and a code editor on the right. The code editor displays the Java code for a 'Dice' class. The code includes an import statement for 'java.util.Random', a class definition 'Dice' with a static variable 'sidesOfDice' and an instance variable 'faceValue'. It also contains three methods: 'roll()' which uses 'Random' to generate a random number, 'changeNumSidesOfDice()' which is a static method that updates the 'sidesOfDice' variable, and 'getFaceValue()' which returns the current 'faceValue'.

## Access Modifiers

Access modifiers determine how certain variables and methods can be accessed

## Access Modifiers

	Class	Package	Subclass (same pkg)	Subclass (diff pkg)	World
public	+	+	+	+	+
protected	+	+	+	+	
no modifier	+	+	+		
private	+				

+ : accessible

blank : not accessible



**String manipulation in Java**

# String: A Definition

- A string is comprised of a set of characters, including letters, numbers, and spaces
- `String s = "abc";`
- `String name = "Shelly Parker";`
- `String identifier = "19Y7W248E"`



1

## String Literals

- `String A = "abc";`
- `String B = "abc";`
- Live in string constant pool
- Strings A and B reference the same object and value

2

## String Objects

- `String A = new String("abc");`
- `String B = new String("abc");`
- Live in the heap
- String A and B reference two separate objects



## Create string literals rather than string objects.

If String A and String B have the same content, then it is nice to just have one place where it is stored (**string literals**) rather than multiple (**string objects**).

Linked 

## Important Facts about Strings

- Strings are immutable in Java
- Classes like `StringBuilder` and `StringBuffer` allow you to make string-like objects that are mutable
- `StringBuffer` is thread-safe because it has synchronized methods

Linked 

## Different ways to concatenate Strings

Solutions for: Business Higher Education

### Watching: Different ways to concatenate strings

From the course: Nail Your Java Interview

2,134 4,036

```
1 public class Main {
2
3     public static void main(String[] args) {
4         String firstName = "Shelly";
5         String lastName = "Parker";
6
7         String name = firstName + " " + lastName;
8
9         System.out.println(name);
10        System.out.println(firstName);
11        System.out.println(lastName);
12    }
13 }
14
```

Run Main

```
/Library/Java/JavaVirtualMachines/jdk-9.0.1.jdk/Contents/Home/bin/java "-javaagent:/Applicat
Shelly Parker
Shelly
Parker

Process finished with exit code 0
```

Compilation completed successfully in 809ms (moments ago)

11:38 LF1 UTF-8

## Data Structure

### Array



## What Is an Array?

- A collection of objects of the same data type
- Size cannot be changed after allocation since items are stored contiguously
- Access items by index, starting at 0
- Ability to add and remove objects from an array



### How to use Linked list

## What Is a Linked List?

- A linear data structure where elements containing data of the same type are linked using pointers
- Each item in the linked list is called a node
- A node contains data, as well as a pointer pointing to the next element in the linked list
- If a next pointer is null, then it is the last item in the list



# Pointers in a Linked List

- Since elements are connected by pointers, they do not need to be stored next to each other
- **Singly-linked lists:** contain a pointer pointing to the next node
- **Doubly-linked lists:** contain a pointer pointing to the next node, and a pointer pointing to the previous node

LinkedIn

## Pointers in a List

1

### Linked List Advantages

- Quicker at inserting and deleting dynamically sized data
- Used to implement stacks and queues

2

### Linked List Disadvantages

- More memory is required for storing elements in a linked list
- Search or node traversal is still time consuming

## What Is a Linked List?

- A linear data structure where elements containing data of the same type are linked using pointers
- Each item in the linked list is called a node
- A node contains data, as well as a pointer pointing to the next element in the linked list

How to use Queue

## What Is a Queue?

- An ordered list of objects where elements are inserted (enqueued) at the end of the queue, and removed (dequeued) from the beginning of the queue
- First in first out —> FIFO

# Queue Interface in Java

- `java.util.Queue` interface
- You must instantiate a concrete implementation of the interface in order to use it
- Linked list is the standard queue implementation

## Using Queues

- Good for storing order of processes
- Enqueue/dequeue takes very little time
- Only advantageous to use this data structure when you want to use it in a manner where the first item in is the first item out

**How to use stack**

## Using Stacks

- A stack is an ordered list of objects that are inserted and removed following a last-in-first-out (LIFO) policy
- Insert items with a `push()` method
- Delete items with a `pop()` method

A stack is an ordered list of objects inserted and removed following a last-in-first-out (LIFO) policy.

## Why Use Stacks

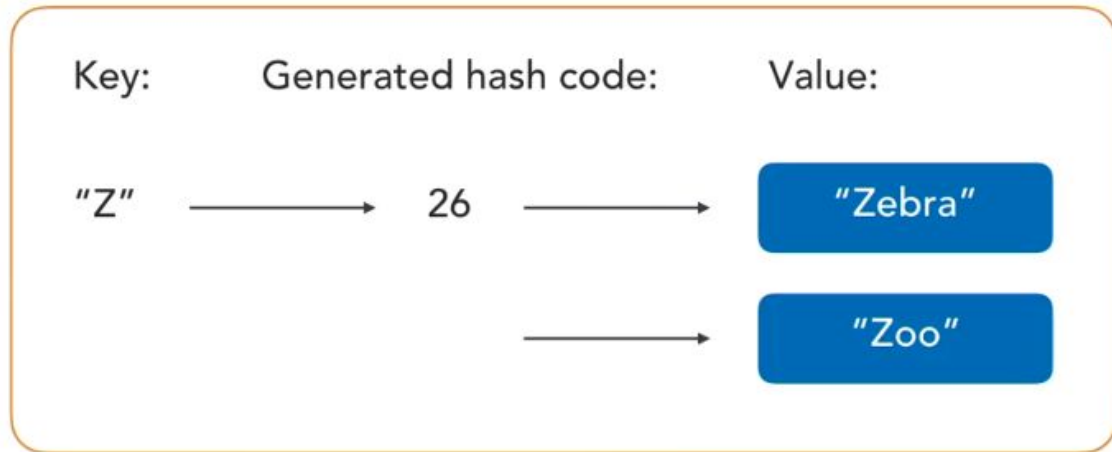
- Great for reversing things
- Imagine pushing all the characters from a string onto a stack—if we pop them all off and create a resulting string, this resulting string has the reversed characters of the original string

### Hash Map

## What are Hash Maps?

- **Hash maps** use key-value pairs and a hashing function to store and organize their data
- A **hashing function** maps a key or object to a specific hash
- This **hash** determines where the object is stored
- As long as you have the key, retrieving the object is fast

## How Hashmaps Work: Hash Collisions



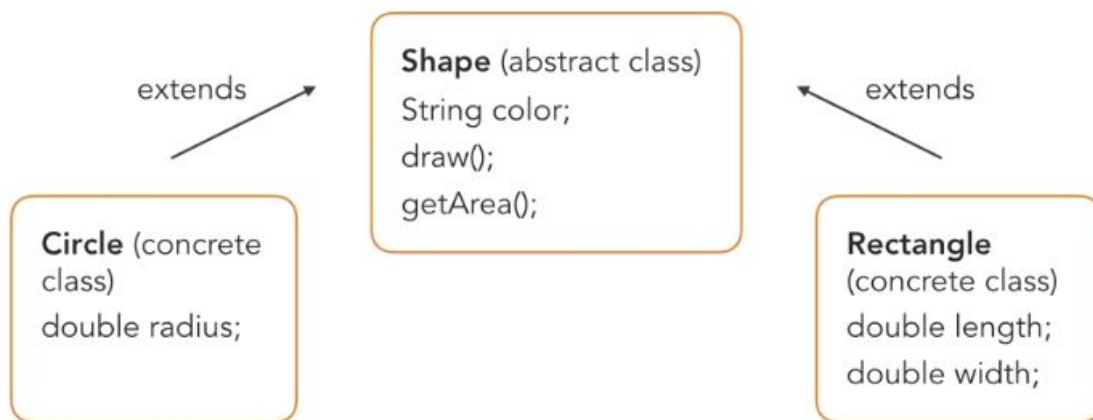
Searching is faster with a hash map,  
but a hash map takes up more space.

## What Is Abstraction?

- Abstraction hides implementation complexity so we can achieve generalization
- All the user needs to know is an example, input, out, and a broad description of what the function does—this allows the function to be used for multiple purposes
- Consider pressing the gas pedal on a car...

LinkedIn

## Abstraction Example



LinkedIn



## Why Use Abstraction?

- Abstraction allows only essential details to be displayed to the user so that the feature can be used in more ways rather than in one super specific way
- It also lets you focus on the required characteristics and ignore irrelevant implementation details, providing generalization to the program



### Encapsulation

## What Is Encapsulation?

- Binding an object's state and behaviors together  
(In Java, fields and methods)
- A way of wrapping data and code acting on the data into a single unit  
(In Java, a class)
- Keeps classes separated and prevents coupling



With data private and methods public, we allow other classes to access hidden data, but they can only do so indirectly with a public method.

## What Is Encapsulation?

- With encapsulation, a class' data is hidden from other classes and can only be accessed through specific methods of its own class—this is called data hiding
- In Java, we achieve encapsulation by declaring **all the fields in a class as private and writing public methods** in the class to set and get the values of variables

## Disadvantages of Inheritance

- The main disadvantage of inheritance is that the superclass and subclass can become tightly coupled, meaning they cannot be used independently of each other
- The program extends increased effort to jump through all the levels of implementation to get to the appropriate functionality



## Compile Time Polymorphism

- **Compile time polymorphism** is achieved through method overloading
- **Method overloading** is a feature that allows a class to have more than one method having the same name, if their argument lists are different
- Again, this is similar to before where one method or function has many different forms

## Why Encapsulation?

- We use encapsulation so that the user has no idea of the inner implementation of a given class and the data it contains
- It allows you to hide how values are stored within a given class

Encapsulation differs from abstraction.

Abstraction provides generalization.

Encapsulation hides unwanted implementation details from the users of an object.

Linked 

**Inheritance**

## What Is Inheritance?

- Inheritance is the process where one class acquires the fields and methods of another
- With inheritance, we can write the common properties and functionality in one class and have other classes with unique features all extend this one class, making code more reusable

## What Is Inheritance?

- Instead of writing the same common implementation multiple times, we can write it once and then have whatever class needs this functionality extend this class

## Why Use Inheritance?

- The main advantage of inheritance is minimizing the amount of duplicate code
- Inheritance allows data hiding, where the super base class can keep some data private, and this data cannot be accessed by the subclass

LinkedIn

### Polymorphism

## What Is Polymorphism?

- Polymorphism is the ability for an object or function to take many forms
- **Runtime polymorphism** is achieved through method overriding
- **Compile time polymorphism** is achieved through method overloading