21 Days — Zero to Confident Java Developer

■ 21 DAYS	■ 120+ Code Examples	■ 18 Projects	■ Real-World Apps

WEEK 1: Core Java	WEEK 2: OOP	WEEK 3: Advanced Java

WEEK 1 — CORE JAVA FUNDAMENTALS

Day 1 — What is Java? Setup & First Program

Day 2 — Variables, Data Types & Operators

Day 3 — Conditionals: if / else / switch

Day 4 — Loops: for / while / do-while

Day 5 — Arrays & Methods

Day 6 — String Operations

Day 7 — WEEK 1 PROJECT — Console Applications

WEEK 2 — OBJECT-ORIENTED PROGRAMMING

Day 8 — Classes & Objects (OOP Basics)

Day 9 — Constructor, this & Encapsulation

Day 10 — Inheritance

Day 11 — Polymorphism & Interface

Day 12 — Abstract Class & Exception Handling

Day 13 — Collections: ArrayList & HashMap

Day 14 — WEEK 2 PROJECT — OOP Application

WEEK 3 — ADVANCED JAVA & REAL WORLD

Day 15 — Generics & Lambda Expressions

Day 16 — Stream API & Functional Programming

Day 17 — File I/O

Day 18 — Algorithms & Big-O Complexity

Day 19 — Database with Java (SQLite / JDBC)

Day 20 — Maven, Build Tools & Clean Code

Day 21 — FINAL PROJECT — Full Application

WEEK 1 | CORE JAVA FUNDAMENTALS

Variables · Conditionals · Loops · Arrays · Methods · Strings

DAY 1 What is Java? Setup & First Program ~3 hrs

■ **Gunun Hedefi:** Java'nin ne oldugunu anlamak, gelistirme ortamini kurmak ve ekrana ilk ciktiyi yazdirmak.

1.1 Java Nedir?

Java, 1995'te Sun Microsystems tarafindan gelistirilen, platform bagimsiz, nesne tabanli bir programlama dilidir. **"Write Once, Run Anywhere (WORA)"** prensibi sayesinde bir kez derlenen kod, JVM (Java Virtual Machine) kurulu her sistemde calisir.

Ozellik	Java	Python	C++
Hiz	Cok hizli (JIT)	Orta	En hizli
Zorluk	Orta	Kolay	Zor
Platform	JVM — her yerde	Interpreter	Derleme gerekir
Kullanim	Backend / Android	AI / Script	Sistem / Oyun
Type System	Static (derleme)	Dynamic (runtime)	Static (derleme)

■ DIKKAT

Java ile JavaScript TAMAMEN farkli dillerdir! Sadece isimleri benzer. Birbirinin versiyonu ya da devami DEGILDIR.

1.2 JDK / JRE / JVM Farki

Kisaltma	Acilimi	Amaci
JVM	Java Virtual Machine	Bytecode'u calistirir (her OS icin ayri)
JRE	Java Runtime Environment	JVM + standart kutuphaneler (sadece calistirma)
JDK	Java Development Kit	JRE + derleyici + araclar (gelistirme icin)

■ BILGI

Gelistirici olarak JDK kurmalisiniz. Sadece Java uygulamasi calistirmak isteyenler JRE ile yetinebilir.

1.3 Kurulum

- **1. JDK Indir:** https://adoptium.net → 'Java 21 LTS' sec (LTS = Long Term Support)
- **2. IDE Kur:** https://www.jetbrains.com/idea/ → 'Community Edition' ucretsiz
- **3. Dogrula:** Terminal / CMD ac, asagidaki komutu calistir:

```Java
java -version
// Beklenen cikti:
// openjdk version "21.x.x" ...
```

```
// OpenJDK Runtime Environment ...
// OpenJDK 64-Bit Server VM ...
```

■ IPUCU

IntelliJ IDEA'da otomatik tamamlama: 'psvm' + Tab = main metodu, 'sout' + Tab = System.out.println!

1.4 Ilk Program — Hello World

Java'da HER sey bir **sinif (class)** icinde olmalidir. Program calismasina **main** metodundan baslar. Dosya adi ile public class adi AYNI olmalidir.

Java

```
// File: HelloWorld.java   <--- filename MUST match class name!
public class HelloWorld {
    // Entry point of every Java program
    public static void main(String[] args) {
        System.out.println("Hello, World!");   // prints + newline
        System.out.print("No newline here ");  // prints, NO newline
        System.out.println("continuing...");
        // Formatted output
        System.out.printf("Name: %s, Age: %d, GPA: %.2f%n", "Alice", 20, 3.85);
    }
}
// Expected output:
// Hello, World!
// No newline here continuing...
// Name: Alice, Age: 20, GPA: 3.85
```

■ DIKKAT

Dosya adi ile public class adi HER ZAMAN ayni olmali ve buyuk/kucuk harfe duyarlidir. 'helloworld.java' ile 'HelloWorld.java' farklidir!

1.5 Derleme ve Calistirma

Java

```
// --- Terminal / Command Prompt ---
// Step 1: Compile  (creates HelloWorld.class = bytecode)
javac HelloWorld.java
// Step 2: Run  (JVM executes the bytecode)
java HelloWorld
// Note: do NOT write 'java HelloWorld.class' — just the class name!
```

1.6 Yorum Satirlari (Comments)

Java

```
// Single-line comment — ignored by compiler
/*
   Multi-line comment
   Can span several lines
*/
/**
 * JavaDoc comment — used to auto-generate documentation
 * @param name  the name to greet
 * @return      the greeting message
 */
public static String greet(String name) {
```

```
    return "Hello, " + name + "!";
}
```

■ PROJECT: Introduce Yourself [Beginner]

➤ Print your name, surname and age — each on a separate line

➤ Use at least one System.out.print and one System.out.println

➤ Use printf to print: 'Hello! My name is [name], I am [age] years old.'

➤ Add a multi-line comment at the top describing what the program does

➤ BONUS: Print a simple ASCII art (e.g. a smiley face) using println

■ ***Bugun ogrendiklerin:*** *JDK/JVM/JRE farki, Java dosya yapisi, derleme vs calistirma, yorum satirlari, print / println / printf.*

DAY 2 **Variables, Data Types & Operators** *~3.5 hrs*

■ **Gunun Hedefi:** Degisken kavramini, 8 primitive veri tipini, wrapper siniflari ve operatorleri ogrenip hesaplama yapan programlar yazmak.

2.1 Variables (Degiskenler)

Degisken, verinin hafizada saklandigi isimli kutudur. Java'da degisken kullanmadan once tipi belirtilmek ZORUNDADIR (statically typed language).

```Java
// Syntax: type variableName = value;
int age = 25;
double height = 1.75;
boolean isStudent = true;
char grade = 'A';         // single quotes for char!
String name = "Alice";  // double quotes for String!
// Declare first, assign later
int score;
score = 100;
// Multiple variables of same type
int x = 1, y = 2, z = 3;
// var keyword (Java 10+) — type inferred by compiler
var count = 42;        // compiler infers: int
var message = "Hi";  // compiler infers: String
// var only works for LOCAL variables (inside methods)
```

■ DIKKAT

char icin tek tirnak ('A'), String icin cift tirnak ("Alice") kullanilir. Karistirirsamz derleme hatasi alirsiniz!

2.2 8 Primitive Data Types

Type	Size	Range / Notes	Example
byte	1 byte	-128 to 127	byte b = 100;
short	2 bytes	-32,768 to 32,767	short s = 2000;
int	4 bytes	-2.1B to 2.1B (default int)	int i = 500_000;
long	8 bytes	Very large numbers	long l = 100L; // L required!
float	4 bytes	~7 decimal digits	float f = 3.14f; // f required!
double	8 bytes	~15 decimal digits (default)	double d = 3.14159;
boolean	1 bit	true or false only	boolean ok = true;
char	2 bytes	Single Unicode character	char c = 'Z';

■ IPUCU

Cogu sayisal islemde int (tam sayi) ve double (ondalikli) yeterlidir. long ve float yalnizca ozel durumlarda gerekir.

2.3 Wrapper Classes (Sarmalayici Siniflar)

Her primitive tipin bir nesne versiyonu (wrapper class) vardir. ArrayList gibi koleksiyonlarda primitive tipler KULLANILAMAZ — wrapper kullanilir.

```java
// Primitive   ->  Wrapper
// int         ->  Integer
// double      ->  Double
// boolean     ->  Boolean
// char        ->  Character
Integer num = 42;       // Autoboxing: int -> Integer automatically
int primitive = num;    // Unboxing:   Integer -> int automatically
// Useful wrapper methods
int max = Integer.MAX_VALUE;     // 2147483647
int min = Integer.MIN_VALUE;     // -2147483648
int parsed = Integer.parseInt("123");     // String -> int
String str = Integer.toString(456);        // int -> String
String bin = Integer.toBinaryString(10);   // "1010"
```

2.4 Operators (Operatorler)

Arithmetic Operators:

```java
int a = 10, b = 3;
System.out.println(a + b);   // 13  addition
System.out.println(a - b);   // 7   subtraction
System.out.println(a * b);   // 30  multiplication
System.out.println(a / b);   // 3   integer division (NOT 3.33!)
System.out.println(a % b);   // 1   modulus (remainder)
System.out.println(a / (double) b); // 3.3333  cast to double first!
// Increment / Decrement
int x = 5;
System.out.println(x++);  // prints 5, THEN increments -> x=6
System.out.println(++x);  // increments first -> x=7, THEN prints 7
System.out.println(x--);  // prints 7, THEN decrements -> x=6
// Compound assignment
x += 3;   // x = x + 3
x -= 2;   // x = x - 2
x *= 4;   // x = x * 4
x /= 2;   // x = x / 2
x %= 3;   // x = x % 3
```

■ DIKKAT

int / int = int sonucu verir! 10 / 3 = 3 yazar, 3.33 DEGILDIR. Ondalikli sonuc icin en az birini double'a cast edin: (double)a / b

Comparison & Logical Operators:

```java
// Comparison operators — always return boolean
System.out.println(5 == 5);    // true   (equal to)
System.out.println(5 != 3);    // true   (not equal to)
System.out.println(5 >  3);    // true   (greater than)
System.out.println(5 <  3);    // false  (less than)
System.out.println(5 >= 5);    // true   (greater or equal)
System.out.println(5 <= 4);    // false  (less or equal)
```

```java
// Logical operators
boolean a = true, b = false;
System.out.println(a && b);   // false  AND  (both must be true)
System.out.println(a || b);   // true   OR   (at least one true)
System.out.println(!a);       // false  NOT  (inverts boolean)
// Short-circuit evaluation
// In (x != 0 && 10/x > 1), if x==0, second part is NOT evaluated!
int x = 0;
if (x != 0 && 10 / x > 1) System.out.println("safe");
// No division by zero error — short-circuit protects us
```

2.5 Scanner — Kullanicidan Girdi Alma

Java

```java
import java.util.Scanner;  // add this at the TOP of the file
public class UserInput {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your name: ");
        String name = scanner.nextLine();   // reads whole line
        System.out.print("Enter your age: ");
        int age = scanner.nextInt();        // reads integer
        System.out.print("Enter your GPA: ");
        double gpa = scanner.nextDouble();  // reads double
        System.out.printf("Hello %s! Age: %d, GPA: %.2f%n", name, age, gpa);
        scanner.close();  // good practice to close
    }
}
```

■ DIKKAT

scanner.nextInt() cagrisindan SONRA scanner.nextLine() kullaniyorsaniz, araya bos bir scanner.nextLine() eklemeniz gerekir. Nedeni: nextInt() Enter tusunu tamponda birakir.

■ PROJECT: Calculator v1.0 [Beginner]

➤ Get 2 numbers from user with Scanner

➤ Perform: addition, subtraction, multiplication, division

➤ Print all 4 results with labels (e.g. '10 + 3 = 13')

➤ Check for division by zero with if statement

➤ BONUS: Calculate and print the power of a to b (use Math.pow)

➤ BONUS: Print whether the sum is even or odd (use % operator)

DAY 3 **Conditionals: if / else / switch** *~3.5 hrs*

■ **Gunun Hedefi:** Program akisini kosula bagli olarak kontrol etmeyi, kararlari if/else ve switch ile ifade etmeyi ogrenin.

3.1 if / else if / else

```java
int score = 75;
if (score >= 90) {
    System.out.println("Grade: A");
} else if (score >= 80) {
    System.out.println("Grade: B");
} else if (score >= 70) {
    System.out.println("Grade: C");
} else if (score >= 60) {
    System.out.println("Grade: D");
} else {
    System.out.println("Grade: F");
}
// Output: Grade: C
// Real-world example: login check
String username = "admin";
String password = "secret123";
if (username.equals("admin") && password.equals("secret123")) {
    System.out.println("Login successful!");
} else if (username.equals("admin")) {
    System.out.println("Wrong password!");
} else {
    System.out.println("User not found!");
}
```

■ DIKKAT

Kosul blogunda tek satir olsa bile suslu parantez {} kullanin! Tek satirlik if hata yapmaya cok aciktir ve okumasi zordur.

3.2 Ternary Operator (Uclu Operator)

Ternary operatoru, basit if-else ifadelerinin kisaltmasidir: **kosul ? dogruysa_bu : yanlissa_bu**

```java
// Syntax: condition ? valueIfTrue : valueIfFalse
int number = 7;
String result = (number % 2 == 0) ? "Even" : "Odd";
System.out.println(result);  // Odd
// Another example: absolute value
int val = -5;
int abs = (val >= 0) ? val : -val;
System.out.println(abs);  // 5
// Max of two numbers
int a = 10, b = 20;
int max = (a > b) ? a : b;
System.out.println("Max: " + max);  // Max: 20
// BAD: nested ternary — avoid this, very unreadable
// String grade = score>=90 ? "A" : score>=80 ? "B" : score>=70 ? "C" : "F";
```

■ IPUCU

Ternary operatoru basit durumlarda if/else'in guzel kisaltmasidir. Ancak icice ternary yazmaktan kacinin — okunaksiz olur.

3.3 switch / case

Java

```
// Classic switch (all Java versions)
int day = 3;
switch (day) {
    case 1: System.out.println("Monday");    break;
    case 2: System.out.println("Tuesday");   break;
    case 3: System.out.println("Wednesday"); break;
    case 4: System.out.println("Thursday");  break;
    case 5: System.out.println("Friday");    break;
    case 6: System.out.println("Saturday");  break;
    case 7: System.out.println("Sunday");    break;
    default: System.out.println("Invalid day");
}
// Multiple values in one case
switch (day) {
    case 1: case 2: case 3: case 4: case 5:
        System.out.println("Weekday"); break;
    case 6: case 7:
        System.out.println("Weekend"); break;
}
```

■ DIKKAT

switch/case icinde break UNUTMAYINIZ! Break unutulursa 'fall-through' olur: eslesme sonrasindaki TUM case'ler de calisir. Bu cok yaygin bir hatadir!

Java

```
// Modern switch expression (Java 14+) — cleaner, no break needed
String dayName = switch (day) {
    case 1 -> "Monday";
    case 2 -> "Tuesday";
    case 3 -> "Wednesday";
    case 4, 5 -> "Thursday or Friday";
    case 6, 7 -> "Weekend";
    default -> "Invalid";
};
System.out.println(dayName);  // Wednesday
// switch also works with String
String season = "summer";
String description = switch (season.toLowerCase()) {
    case "spring" -> "Flowers bloom";
    case "summer" -> "Hot and sunny";
    case "autumn" -> "Leaves fall";
    case "winter" -> "Cold and snowy";
    default -> "Unknown season";
};
```

Durum	if / else	switch

Aralik kontrolu (>60, <=90)	Kullan	Kullanilamaz
Sabit deger (1,2,3)	Kullanabilir	Daha temiz
boolean kosullar	Kullan	Uygun degil
String degeri kontrolu	Kullanabilir	Java 7+ destekler
Birden fazla sabit esitlik	Daha uzun	Ideal

■ PROJECT: Grade Calculator & Menu System [Beginner]

➤ Get 5 subject scores from user

➤ Calculate average and print letter grade (A/B/C/D/F)

➤ Build a calculator menu: 1=Add, 2=Sub, 3=Mul, 4=Div, 0=Exit

➤ Use switch to handle menu, if for division-by-zero guard

➤ Print 'PASS' if average >= 60, 'FAIL' otherwise (use ternary)

➤ BONUS: Check if a number is positive, negative, or zero

➤ BONUS: Check if a year is a leap year

DAY 4 **Loops: for / while / do-while** *~4 hrs*

■ **Gunun Hedefi:** Tekrarli islemleri dongulerle yazmak, break/continue kullanmak ve ic ice donguleri anlamak.

4.1 for Loop

```java
// Syntax: for (initialization; condition; update)
for (int i = 0; i < 5; i++) {
    System.out.println("Step: " + i);
}
// 0, 1, 2, 3, 4  (NOT 5!)
// Count backwards
for (int i = 10; i >= 1; i--) {
    System.out.print(i + " ");
}
// 10 9 8 7 6 5 4 3 2 1
// Step by 2
for (int i = 0; i <= 20; i += 2) {
    System.out.print(i + " ");
}
// 0 2 4 6 8 10 12 14 16 18 20
// Sum of 1 to 100
int sum = 0;
for (int i = 1; i <= 100; i++) {
    sum += i;
}
System.out.println("Sum 1-100: " + sum);  // 5050
```

■ DIKKAT

i < 5 ile i <= 5 arasindaki farka dikkat! i<5 → 0,1,2,3,4 (5 deger); i<=5 → 0,1,2,3,4,5 (6 deger). Off-by-one hatasi cok yaygindir!

4.2 while Loop

```java
// Condition checked BEFORE each iteration
// Use when you DON'T know how many times to loop
int i = 0;
while (i < 5) {
    System.out.println("i = " + i);
    i++;   // NEVER FORGET THIS! Infinite loop otherwise
}
// Real use case: keep asking until valid input
Scanner scanner = new Scanner(System.in);
int number = -1;
while (number < 1 || number > 100) {
    System.out.print("Enter a number between 1 and 100: ");
    number = scanner.nextInt();
    if (number < 1 || number > 100) {
        System.out.println("Invalid! Try again.");
    }
}
System.out.println("You entered: " + number);
```

```java
// Loop until user types 'quit'
String input = "";
while (!input.equalsIgnoreCase("quit")) {
    System.out.print("Type something (quit to exit): ");
    input = scanner.nextLine();
    System.out.println("You said: " + input);
}
```

■ HATA

Dongude artirmayi (i++) UNUTMAK = SONSUZ DONGU! Program donar. IntelliJ'de Ctrl+C ile durdurabilirsiniz.

4.3 do-while Loop

Kod ONCE calisir, SONRA kosul kontrol edilir. En az 1 kez calisma garantilidir. Menulerde ve input dogrulamasinda idealdir.

Java

```java
// Condition checked AFTER each iteration
// Guaranteed to run at least ONCE
int choice;
Scanner scanner = new Scanner(System.in);
do {
    System.out.println("\n--- MENU ---");
    System.out.println("1. Start Game");
    System.out.println("2. Settings");
    System.out.println("0. Exit");
    System.out.print("Your choice: ");
    choice = scanner.nextInt();
    switch (choice) {
        case 1 -> System.out.println("Game started!");
        case 2 -> System.out.println("Opening settings...");
        case 0 -> System.out.println("Goodbye!");
        default -> System.out.println("Invalid option.");
    }
} while (choice != 0);
```

4.4 break ve continue

Java

```java
// break: exits the loop entirely
for (int i = 0; i < 10; i++) {
    if (i == 5) break;
    System.out.print(i + " ");
}
// Output: 0 1 2 3 4
// continue: skips the current iteration
for (int i = 0; i < 10; i++) {
    if (i % 2 == 0) continue;  // skip evens
    System.out.print(i + " ");
}
// Output: 1 3 5 7 9
// Real use case: find first prime > 10
int n = 11;
while (true) {  // infinite loop, break when done
    boolean isPrime = true;
```

```
    for (int i = 2; i <= Math.sqrt(n); i++) {
        if (n % i == 0) { isPrime = false; break; }
    }
    if (isPrime) { System.out.println("First prime > 10: " + n); break; }
    n++;
}
// Output: First prime > 10: 11
```

4.5 Nested Loops & Patterns

Java

```
// Multiplication table
System.out.println("--- Multiplication Table ---");
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 5; j++) {
        System.out.printf("%4d", i * j);
    }
    System.out.println();
}
// Star triangle
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print("* ");
    }
    System.out.println();
}
// *
// * *
// * * *
// * * * *
// * * * * *
// Diamond pattern
int n = 5;
for (int i = 1; i <= n; i++) {
    for (int j = i; j < n; j++) System.out.print(" ");
    for (int j = 1; j <= 2*i-1; j++) System.out.print("*");
    System.out.println();
}
```

■ PROJECT: Number Games [Beginner]

- ➤ FizzBuzz: print 1-100, multiples of 3='Fizz', 5='Buzz', both='FizzBuzz'
- ➤ Prime Finder: list all primes from 1 to 100
- ➤ Guessing Game: generate random 1-100, user guesses with hints (higher/lower)
- ➤ Print multiplication table (10x10) using printf for alignment
- ➤ Fibonacci: print first 20 terms of the Fibonacci sequence
- ➤ BONUS: Find all perfect numbers between 1 and 10000
- ➤ BONUS: Collatz conjecture — pick a number, apply rules, count steps to reach 1

DAY 5 **Arrays & Methods** *~4 hrs*

■ **Gunun Hedefi:** Birden fazla veriyi dizi ile saklamak, kendi metotlarini yazmak ve Arrays sinifini kullanmak.

5.1 Arrays (Diziler)

Java

```
// Declaration and initialization
int[] numbers = new int[5];           // 5 elements, all 0 by default
int[] scores = {85, 90, 72, 88, 65};  // inline initialization
String[] names = new String[3];        // all null by default
// Access — ZERO-BASED INDEXING!
System.out.println(scores[0]);         // 85  (first element)
System.out.println(scores[4]);         // 65  (last element)
System.out.println(scores.length);     // 5   (not 4!)
// Modify
scores[2] = 95;
// Iterate with for loop
for (int i = 0; i < scores.length; i++) {
    System.out.println("scores[" + i + "] = " + scores[i]);
}
// Enhanced for-each (no index access needed)
int total = 0;
for (int score : scores) {
    total += score;
}
System.out.println("Average: " + (double) total / scores.length);
```

■ **DIKKAT**

Dizi siniri disina cikarsaniz ArrayIndexOutOfBoundsException alirsiniz! 5 elemanli dizide scores[5] → HATA. Son gecerli index: length - 1

5.2 Arrays Utility Class

Java

```
import java.util.Arrays;
int[] nums = {5, 2, 8, 1, 9, 3, 7, 4, 6};
// Print array content
System.out.println(Arrays.toString(nums));  // [5, 2, 8, 1, 9, 3, 7, 4, 6]
// Sort ascending
Arrays.sort(nums);
System.out.println(Arrays.toString(nums));  // [1, 2, 3, 4, 5, 6, 7, 8, 9]
// Binary search (array MUST be sorted first!)
int idx = Arrays.binarySearch(nums, 7);
System.out.println("Found 7 at index: " + idx);  // 6
// Copy
int[] copy = Arrays.copyOf(nums, 5);           // first 5 elements
int[] range = Arrays.copyOfRange(nums, 2, 6);  // elements [2..5]
// Fill
int[] zeros = new int[5];
Arrays.fill(zeros, 0);
// Compare
System.out.println(Arrays.equals(nums, copy));  // false (different length)
```

5.3 2D Arrays

```java
// 2D array = array of arrays (like a table/matrix)
int[][] matrix = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
System.out.println(matrix[1][2]);  // 6  (row 1, col 2)
System.out.println(matrix.length);    // 3  (rows)
System.out.println(matrix[0].length); // 3  (cols)
// Print matrix
for (int r = 0; r < matrix.length; r++) {
    for (int c = 0; c < matrix[r].length; c++) {
        System.out.printf("%3d", matrix[r][c]);
    }
    System.out.println();
}
```

5.4 Writing Methods (Metot Yazmak)

```java
public class MathUtils {
    // returnType methodName(parameters)
    public static int add(int a, int b) {
        return a + b;
    }
    // void: returns nothing
    public static void printLine(String text) {
        System.out.println("--- " + text + " ---");
    }
    // Method overloading: same name, different parameters
    public static double add(double a, double b) {
        return a + b;
    }
    public static int add(int a, int b, int c) {
        return a + b + c;
    }
    // Recursive method
    public static int factorial(int n) {
        if (n <= 1) return 1;          // base case
        return n * factorial(n - 1);   // recursive call
    }
    public static void main(String[] args) {
        System.out.println(add(3, 5));        // 8
        System.out.println(add(1.5, 2.5));    // 4.0
        System.out.println(add(1, 2, 3));     // 6
        System.out.println(factorial(5));     // 120
        printLine("Hello");
    }
}
```

■ DIKKAT

Java'da metotlara primitive tipler KOPYALANARAK gecirilir (pass by value). Metodun icinde int parametresini degistirmek, disaridaki degiskeni ETKILEMEZ. Ancak diziler (array) referans olarak gecirilir — disaridakini degistirir!

■ PROJECT: Array Statistics App [Beginner]

➤ Get 10 numbers from user, store in array

➤ Write methods: sum(), average(), max(), min(), countEvens(), countOdds()

➤ Print sorted array (ascending and descending)

➤ Write a method to search for a value (linear search)

➤ Write a method to reverse an array WITHOUT using a second array

➤ BONUS: Write mergeArrays(int[] a, int[] b) method

➤ BONUS: Find the second largest number in array

DAY 6 **String Operations** *~3.5 hrs*

■ **Gunun Hedefi:** String sinifinin metodlarini, String karsilastirma kuralini, String.format ve StringBuilder'i ogrenin.

6.1 String Temel Metodlar

```Java
String s = "Hello, Java World!";
System.out.println(s.length());              // 18
System.out.println(s.charAt(7));             // J
System.out.println(s.indexOf("Java"));     // 7
System.out.println(s.lastIndexOf('o'));       // 15
System.out.println(s.substring(7, 11));      // Java
System.out.println(s.substring(7));          // Java World!
System.out.println(s.toUpperCase());         // HELLO, JAVA WORLD!
System.out.println(s.toLowerCase());         // hello, java world!
System.out.println(s.replace("Java", "Python"));  // Hello, Python World!
System.out.println(s.contains("Java"));    // true
System.out.println(s.startsWith("Hello")); // true
System.out.println(s.endsWith("!"));       // true
System.out.println(s.isEmpty());             // false
System.out.println(s.isBlank());             // false (Java 11+)
// Trim whitespace
String padded = "   hello   ";
System.out.println(padded.trim());    // "hello"  (trims both ends)
System.out.println(padded.strip());   // "hello"  (Unicode-aware, Java 11+)
```

6.2 String == vs .equals() — KRITIK!

```Java
// String literals: stored in String Pool
String a = "Java";
String b = "Java";
String c = new String("Java");  // forces a NEW object on heap
System.out.println(a == b);          // true  (same pool reference)
System.out.println(a == c);          // FALSE (different object!)
System.out.println(a.equals(c));     // true  (same CONTENT)
System.out.println(a.equalsIgnoreCase("JAVA"));  // true
// RULE: ALWAYS use .equals() for String comparison, NEVER ==
// Bad code (common mistake):
Scanner sc = new Scanner(System.in);
String answer = sc.nextLine();
if (answer == "yes") {  // WRONG! This is almost always false!
    System.out.println("Yes!");
}
// Good code:
if (answer.equals("yes")) {  // CORRECT
    System.out.println("Yes!");
}
```

■ HATA

String karsilastirmasinda == KULLANMAYIN! Bu Java'nin en sik yapilan hatasindan biridir. Her zaman .equals() veya .equalsIgnoreCase() kullanin.

6.3 String Formatting

```Java
String name = "Alice";
int age = 25;
double gpa = 3.875;
// Method 1: concatenation (avoid in loops!)
String s1 = "Name: " + name + ", Age: " + age;
// Method 2: String.format
String s2 = String.format("Name: %s, Age: %d, GPA: %.2f", name, age, gpa);
// Name: Alice, Age: 25, GPA: 3.88
// Method 3: formatted() — Java 15+
String s3 = "Name: %s, Age: %d".formatted(name, age);
// Format specifiers
// %s  = String       %d  = integer       %f  = float/double
// %c  = char         %b  = boolean        %n  = newline
// %.2f = 2 decimal places
// %10s = right-align in 10 chars
// %-10s = left-align in 10 chars
System.out.printf("%-15s %5d  %6.2f%n", name, age, gpa);
```

6.4 StringBuilder (Performance!)

```Java
// BAD: + in loop creates thousands of String objects (SLOW!)
String result = "";
for (int i = 0; i < 10000; i++) {
    result += i;  // Creates a new String each time!
}
// GOOD: StringBuilder — mutable, no extra objects
StringBuilder sb = new StringBuilder();
for (int i = 0; i < 10000; i++) {
    sb.append(i);
}
String result2 = sb.toString();
// StringBuilder methods
StringBuilder builder = new StringBuilder("Hello");
builder.append(" World");        // Hello World
builder.insert(5, ",");           // Hello, World
builder.replace(7, 12, "Java");  // Hello, Java
builder.delete(5, 6);             // Hello Java
builder.reverse();                // avaJ olleH
System.out.println(builder.length());   // 10
System.out.println(builder.toString()); // avaJ olleH
```

■ IPUCU

Dongu icinde String birlestirme icin + yerine StringBuilder kullanin. Her + yeni bir String nesnesi olusturur — buyuk veriler icin cok yavaslar!

6.5 String Splitting & Conversion

```Java
// split()
String csv = "Alice,Bob,Charlie,Diana";
String[] names = csv.split(",");
for (String n : names) System.out.println(n);
```

```java
// String <-> Number conversions
int num    = Integer.parseInt("42");
double d   = Double.parseDouble("3.14");
String str = String.valueOf(123);
String str2 = Integer.toString(456);
// Check if String is numeric
String input = "123abc";
try {
    int n = Integer.parseInt(input);
} catch (NumberFormatException e) {
    System.out.println("Not a valid number!");
}
// char array
char[] chars = "Hello".toCharArray();
String back  = new String(chars);
```

■ PROJECT: Text Processing App [Beginner]

➤ Get a sentence from user

➤ Count: total chars, chars without spaces, words, vowels

➤ Reverse the sentence word by word

➤ Convert to Title Case (first letter of each word uppercase)

➤ Check if it is a palindrome (ignoring spaces and case)

➤ Count how many times a given word appears in the sentence

➤ BONUS: Replace every vowel with '*' and print the censored string

➤ BONUS: Check if two strings are anagrams of each other

DAY 7

WEEK 1 PROJECT — Console Applications

~5 hrs

■ **Gunun Hedefi:** Haftanin tum konularini birlestiren 3 buyuk konsol uygulamasi gelistirmek. Bugun ogretici icerik yok — sadece KOD YAZ!

Week 1 Summary

Topic	Key Point	Common Mistake
Variables	8 primitive types, var keyword	int/int division -> int
Conditionals	if/else, switch, ternary	Forget break in switch
Loops	for/while/do-while, break/continue	Off-by-one, infinite loop
Arrays	0-indexed, .length property	ArrayIndexOutOfBoundsException
Methods	return type, overloading, recursion	Primitive pass-by-value
String	.equals(), format, StringBuilder	Using == for comparison

■ PROJECT: PROJECT 1: Student Grade System [Intermediate]

➤ Store 5 students: name, 4 subject scores

➤ Calculate each student's average

➤ Find class average, highest average, lowest average

➤ Print sorted ranking (highest to lowest average)

➤ Show grade distribution: how many A/B/C/D/F

➤ Format output nicely using printf (aligned columns)

■ PROJECT: PROJECT 2: ATM Simulator [Intermediate]

➤ Starting balance: $5000

➤ Menu: 1=Check Balance, 2=Deposit, 3=Withdraw, 4=Transaction History, 0=Exit

➤ Prevent overdraft (balance < 0)

➤ Show balance after each transaction

➤ Store last 5 transactions in an array

➤ Force valid menu choice with do-while

➤ BONUS: Add 3-attempt PIN lock (PIN = 1234)

■ PROJECT: PROJECT 3: Word Guessing Game (Hangman) [Intermediate]

➤ Store 10 words in a String array

➤ Select a random word: (int)(Math.random() * words.length)

➤ Show word as underscores: _ _ _ _ _

➤ Player guesses one letter at a time

➤ Show correctly guessed letters in their positions

➤ 7 wrong guesses = game over

➤ Track guessed letters, refuse duplicates

➤ BONUS: Show ASCII art of hangman progress

■ IPUCU

Projeleri yaparken once algoritmani kagida yaz (pseudocode). Sonra kucuk adimlarla koda donustur. Tek seferde buyuk kod yazmaya calismak hata yaratir.

■ DIKKAT

Hata aldiginizda panik yapmayin! Hata mesajini oku, hangi satirda oldugunu bul, Google'da ara. Hata okumak programciliginin 40%%'idir.

■ ***Hafta 1 Tamamlandi!*** *Artik Java'nin temel yapi taslarini biliyorsun. Hafta 2'de nesne tabanli programlama ile kodu gercek dunya nesneleri seklinde modellemeyi ogreneceksin.*

WEEK 2 | OBJECT-ORIENTED PROGRAMMING

Classes · Inheritance · Polymorphism · Interfaces · Collections

DAY 8 Classes & Objects — OOP Basics *~4 hrs*

■ **Gunun Hedefi:** Nesne tabanli programlamanin (OOP) temel kavramlarini anlamak ve ilk sinifini tasarlamak.

8.1 OOP Neden Gerekli?

Gercek dunyada her sey bir nesnedir: araba, ogrenci, banka hesabi... OOP bu gercek dunya modelini koda yansitir. Ozellikler **field (alan)**, davranislar **method (metot)** olarak kodlanir. OOP'un 4 temel prensibi vardur:

Prensip	Anlami	Anahtar Kelime
Encapsulation	Veriyi gizle, kontrollü eri■	private, getter/setter
Inheritance	Üst sinifin özelliklerini devral	extends, super
Polymorphism	Ayni arayüz, farkli davranis	@Override, casting
Abstraction	Karmasikli■i gizle	abstract, interface

8.2 First Class Design

Java

```
// File: Student.java
public class Student {
    // Fields (instance variables) — each object has its own copy
    String name;
    String lastName;
    int id;
    double gpa;
    // Method — behavior of the object
    public void displayInfo() {
        System.out.printf("[%d] %s %s | GPA: %.2f%n",
                          id, name, lastName, gpa);
    }
    public String getStatus() {
        if (gpa >= 3.5) return "Honor Roll";
        if (gpa >= 2.0) return "Passing";
        return "Academic Probation";
    }
    public void applyBonus(double bonus) {
        gpa = Math.min(4.0, gpa + bonus);
    }
}
// File: Main.java
public class Main {
    public static void main(String[] args) {
        // 'new' keyword creates an object on the heap
        Student s1 = new Student();
        s1.name = "Alice";
        s1.lastName = "Johnson";
        s1.id = 1001;
```

```
        s1.gpa = 3.75;
        Student s2 = new Student();
        s2.name = "Bob";
        s2.lastName = "Smith";
        s2.id = 1002;
        s2.gpa = 2.3;
        s1.displayInfo();  // [1001] Alice Johnson | GPA: 3.75
        s2.displayInfo();  // [1002] Bob Smith | GPA: 2.30
        System.out.println(s1.getStatus());  // Honor Roll
        s1.applyBonus(0.5);
        s1.displayInfo();  // GPA capped at 4.0
    }
}
```

■ IPUCU

Her public sinif ayri bir .java dosyasinda olmalidir. Proje buyudukce her sinifi kendi dosyasina koymak kodu duzenlemeye yardimci olur.

8.3 Reference vs Primitive

Java

```
// Primitive: value is COPIED
int x = 5;
int y = x;   // y gets a copy
y = 10;
System.out.println(x);  // still 5 — unchanged!
// Object: REFERENCE is copied (both point to same object!)
Student a = new Student();
a.name = "Alice";
Student b = a;   // b points to THE SAME object!
b.name = "Bob";
System.out.println(a.name);  // "Bob" — a changed too!
// To get a true copy, you need to create a new object:
Student c = new Student();
c.name = a.name;  // manually copy each field
c.gpa  = a.gpa;
```

■ DIKKAT

Nesne referanslari birbirini gosterebilir! b = a demek kopya degil, ikisi de ayni nesneyi isaret eder. Gercek kopya icin manuel kopyalama yapilmalidir.

8.4 null & NullPointerException

Java

```
Student s = null;  // s points to NOTHING
// This will crash with NullPointerException!
// System.out.println(s.name);
// Always check for null before using:
if (s != null) {
    System.out.println(s.name);
} else {
    System.out.println("Student not initialized!");
}
// Or use Objects.requireNonNull in constructors:
import java.util.Objects;
```

```
Objects.requireNonNull(s, "Student cannot be null");
```

■ HATA

NullPointerException Java'nin EN yaygin hatasidir! Nesne degiskenine erismeden once null olmadigini kontrol edin.

■ PROJECT: Bank Account Class [Intermediate]

➤ Create BankAccount class: ownerName, accountNumber, balance fields

➤ Methods: deposit(amount), withdraw(amount), getBalance(), printStatement()

➤ Prevent negative balance in withdraw()

➤ Create 3 different account objects and test all methods

➤ Write a transfer(BankAccount target, double amount) method

➤ BONUS: Auto-generate accountNumber using a static counter

➤ BONUS: Track transaction history in a String array (last 10)

DAY 9

Constructor, this & Encapsulation

~4 hrs

■ **Gunun Hedefi:** Constructor ile nesne baslatmayi ve encapsulation (kapsulleme) ile veri guvenligini saglamayi ogrenin.

9.1 Constructor

Constructor, nesne olusturuldugunda otomatik calisir. Sinif adiyla ayni ismi tasir ve donus tipi yoktur. Hicbir constructor yazmazsan Java 'default constructor' ekler.

```Java
public class Car {
    String brand;
    String model;
    int year;
    double price;
    // No-arg constructor
    public Car() {
        brand = "Unknown";
        year  = 2024;
    }
    // Parameterized constructor
    public Car(String brand, String model, int year, double price) {
        this.brand = brand;   // 'this' refers to the current object
        this.model = model;   // distinguishes field from parameter
        this.year  = year;
        this.price = price;
    }
    // Constructor overloading
    public Car(String brand, String model) {
        this(brand, model, 2024, 0.0);  // delegate to full constructor
    }
    @Override
    public String toString() {
        return String.format("%d %s %s ($%.0f)", year, brand, model, price);
    }
}
// Usage
Car c1 = new Car();
Car c2 = new Car("Toyota", "Corolla", 2022, 25000);
Car c3 = new Car("Honda", "Civic");  // overloaded
System.out.println(c2);  // 2022 Toyota Corolla ($25000)
```

■ DIKKAT

this.field = parameter kalibinda this.brand = brand ifadesinde this.brand sinifin alanini, sagdaki brand parametreyi ifade eder. Ikisi ayni isimdeyse this kullanmak ZORUNLUDUR!

9.2 Encapsulation — private + getter/setter

```Java
public class Person {
    private String name;       // outsiders CANNOT access directly
    private int age;
    private String email;
    public Person(String name, int age, String email) {
```

```
        this.name = name;
        setAge(age);           // use setter for validation
        setEmail(email);
    }
    // Getters — read access
    public String getName()  { return name; }
    public int getAge()      { return age; }
    public String getEmail() { return email; }
    // Setters — with validation!
    public void setName(String name) {
        if (name != null && !name.isBlank())
            this.name = name;
    }
    public void setAge(int age) {
        if (age > 0 && age < 150)
            this.age = age;
        else
            throw new IllegalArgumentException("Invalid age: " + age);
    }
    public void setEmail(String email) {
        if (email != null && email.contains("@"))
            this.email = email;
        else
            throw new IllegalArgumentException("Invalid email");
    }
}
// Usage
Person p = new Person("Alice", 25, "alice@example.com");
System.out.println(p.getAge());    // 25
p.setAge(300);  // throws IllegalArgumentException
```

■ IPUCU

IntelliJ IDEA'da getter/setter otomatik uretme: Sinif icinde sag tik -> Generate -> Getter and Setter. Tum private alanlar icin aninda olusturur!

9.3 static Fields and Methods

Java

```
public class Counter {
    private static int count = 0;   // shared by ALL objects
    private int id;
    private String label;
    public Counter(String label) {
        count++;            // increment class-level counter
        this.id    = count;
        this.label = label;
    }
    public static int getCount() { return count; }  // static method
    public int getId()   { return id; }
    public String info() { return "[" + id + "] " + label; }
    // static utility method (no 'this')
    public static int max(int a, int b) { return a > b ? a : b; }
}
Counter c1 = new Counter("First");   // count = 1
Counter c2 = new Counter("Second");  // count = 2
Counter c3 = new Counter("Third");   // count = 3
```

```
System.out.println(Counter.getCount());  // 3  (called on CLASS, not instance)
System.out.println(c1.info());           // [1] First
```

■ PROJECT: Product Catalog (Encapsulation Focus) [Intermediate]

➤ Product class: private id, name, price, stock — full encapsulation

➤ Validate: price > 0, stock >= 0 in setters

➤ ShoppingCart class: Product array, add(), remove(), total() methods

➤ Find most expensive and cheapest product

➤ Print formatted receipt with product names, quantities, subtotals

➤ BONUS: applyDiscount(double percentage) method on Product

➤ BONUS: Static factory method Product.create(name, price)

DAY 10 **Inheritance** *~4 hrs*

■ **Gunun Hedefi:** Kalitim ile kod tekrarini onlemeyi, extends ve super kullanimini ve metot override'ini ogrenin.

10.1 Inheritance Kavrami

Kalitim (inheritance), bir sinifin (child) baska bir sinifin (parent) ozelliklerini ve davranislarini miras almasidir. **"is-a"** iliskisidir: Dog IS-A Animal.

```Java
// Animal.java — Parent class (superclass)
public class Animal {
    private String name;
    private int age;
    private String species;
    public Animal(String name, int age, String species) {
        this.name    = name;
        this.age     = age;
        this.species = species;
    }
    public void makeSound() {
        System.out.println(name + " makes a sound");
    }
    public void eat(String food) {
        System.out.println(name + " eats " + food);
    }
    public String getName()    { return name; }
    public int getAge()        { return age; }
    public String getSpecies() { return species; }
    @Override
    public String toString() {
        return String.format("%s (Species: %s, Age: %d)", name, species, age);
    }
}
// Dog.java — Child class (subclass)
public class Dog extends Animal {
    private String breed;
    private boolean trained;
    public Dog(String name, int age, String breed) {
        super(name, age, "Canis familiaris");  // MUST be first line!
        this.breed   = breed;
        this.trained = false;
    }
    @Override   // overrides parent's makeSound()
    public void makeSound() {
        System.out.println(getName() + " barks: Woof woof!");
    }
    // Dog-specific method
    public void fetch(String item) {
        System.out.println(getName() + " fetches the " + item);
    }
    public void train() { trained = true; System.out.println(getName() + " is now trained!");
 }
    public boolean isTrained() { return trained; }
    public String getBreed()   { return breed; }
```

```
    @Override
    public String toString() {
        return super.toString() + " [Breed: " + breed + ", Trained: " + trained + "]";
    }
}
```

■ DIKKAT

super() cagrisi constructor'in ILK satiri olmalidir! Aksi halde derleme hatasi alirsiniz. super ile this cagrisi ayni constructor'da birlikte kullanilamaz.

10.2 Inheritance Chain & Object Class

Java

```
// Java'da TUM siniflar Object'ten turemistir:
// Object -> Animal -> Dog
Dog d = new Dog("Rex", 3, "German Shepherd");
System.out.println(d);         // calls toString() -> Dog's version
System.out.println(d.getClass().getName());  // Dog
System.out.println(d instanceof Dog);     // true
System.out.println(d instanceof Animal);  // true (IS-A relationship!)
System.out.println(d instanceof Object);  // true (everything is Object)
// Upcasting: Child -> Parent reference (automatic)
Animal a = new Dog("Buddy", 2, "Labrador");  // OK!
a.makeSound();   // calls Dog's makeSound() — polymorphism!
// a.fetch("ball");  // ERROR! Animal reference can't see Dog methods
// Downcasting: Parent -> Child reference (manual, careful!)
if (a instanceof Dog) {
    Dog dog = (Dog) a;  // safe because we checked first
    dog.fetch("ball"); // now we can call Dog methods
}
// Java 16+ pattern matching — cleaner
if (a instanceof Dog dog) {
    dog.fetch("ball");  // dog variable available here
}
```

■ PROJECT: Animal Farm Simulation [Intermediate]

- ➤ Animal (parent), Dog, Cat, Bird, Horse, Fish subclasses
- ➤ Each animal overrides makeSound(), eat(), toString()
- ➤ Farm class: Animal array (max 20), methods: add, remove, listAll
- ➤ Loop through all animals and call makeSound() polymorphically
- ➤ Find the oldest animal, count by species
- ➤ BONUS: Add feeding schedule: each animal eats different food at different times
- ➤ BONUS: Implement Animal.compareTo() to sort by age

DAY 11 **Polymorphism & Interface** *~4.5 hrs*

■ **Gunun Hedefi:** Polimorfizm kavramini, interface ile coklu davranis tanimlama ve upcasting/downcasting'i ogrenin.

11.1 Polymorphism

```Java
// Same method call — different behavior based on actual object type
Animal[] animals = {
    new Dog("Rex", 3, "Shepherd"),
    new Cat("Whiskers", 2),
    new Bird("Tweety", 1),
    new Dog("Buddy", 5, "Labrador")
};
// One loop, four different behaviors!
for (Animal animal : animals) {
    animal.makeSound();   // each calls its own overridden version
}
// Rex barks: Woof woof!
// Whiskers meows: Meow!
// Tweety chirps: Tweet tweet!
// Buddy barks: Woof woof!
// Counting types with instanceof
int dogs = 0, cats = 0;
for (Animal a : animals) {
    if (a instanceof Dog) dogs++;
    else if (a instanceof Cat) cats++;
}
System.out.println("Dogs: " + dogs + ", Cats: " + cats);
```

11.2 Interface

Interface yalnizca **ne yapilacagini** tanimlar, **nasil yapilacagini** degil. Bir sinif birden fazla interface uygulayabilir — kalitimda bu mumkun degildir.

```Java
// Flyable.java
public interface Flyable {
    double MAX_ALTITUDE = 10000;  // automatically: public static final
    void fly();                   // abstract method (no body)
    void land();
    // default method (Java 8+) — has a body, can be overridden
    default void checkFuel() {
        System.out.println("Checking fuel levels...");
    }
    // static method — called on interface itself
    static String info() {
        return "Flyable: max altitude " + MAX_ALTITUDE + "m";
    }
}
// Swimmable.java
public interface Swimmable {
    void swim();
    default void dive(int meters) {
        System.out.println("Diving " + meters + " meters");
```

```
    }
}
// Duck.java — extends Animal AND implements multiple interfaces
public class Duck extends Animal implements Flyable, Swimmable {
    public Duck(String name) { super(name, 1, "Duck"); }
    @Override public void makeSound() { System.out.println(getName() + ": Quack!"); }
    @Override public void fly()  { System.out.println(getName() + " is flying!"); }
    @Override public void land() { System.out.println(getName() + " landed."); }
    @Override public void swim() { System.out.println(getName() + " is swimming!"); }
}
// Using interface as type
Flyable f = new Duck("Donald");
f.fly();
f.checkFuel();
System.out.println(Flyable.info());
```

11.3 Abstract Class vs Interface

Ozellik	Abstract Class	Interface
Coklu miras	NO — tek ust sinif	YES — birden fazla
Constructor	YES — olabilir	NO — olamaz
Field	Her turlu alan	Sadece public static final
Method body	abstract veya normal	default/static haric yok
Access modifiers	public/protected/private	Hepsi public (implicitly)
Ne zaman?	"is-a" iliskisi + ortak kod	"can-do" davranis sozlesmesi

■ DIKKAT

Abstract class tek kalitima izin verir, interface coguna. Tasarimda 'is-a' icin abstract class, 'can-do' yetkinlikler icin interface kullan.

■ PROJECT: Vehicle Rental System [Intermediate]

➤ Rentable interface: rent(int days), returnVehicle(), double getRentalCost()

➤ Refuelable interface: refuel(), double getFuelCost()

➤ Vehicle (abstract): brand, model, dailyRate, abstract calculateRent(int days)

➤ Sedan, SUV, ElectricCar, Motorcycle — all extend Vehicle + implement interfaces

➤ Rental agency: manage fleet, process rentals, calculate revenue

➤ Find cheapest vehicle for X days, most rented vehicle

➤ BONUS: Electric car: calculateRent() includes charging cost

DAY 12 **Abstract Class & Exception Handling** *~4 hrs*

■ **Gunun Hedefi:** Abstract siniflari ve hata yonetimini (exception handling) ogrenip saglam, cokenez programlar yazmak.

12.1 Abstract Class

```java
public abstract class Shape {
    private String color;
    private boolean filled;
    public Shape(String color, boolean filled) {
        this.color  = color;
        this.filled = filled;
    }
    // Abstract methods — MUST be implemented by subclasses
    public abstract double getArea();
    public abstract double getPerimeter();
    public abstract String getDescription();
    // Concrete method — shared behavior
    public void printInfo() {
        System.out.printf("%s Shape | Color: %s | Area: %.2f | Perimeter: %.2f%n",
                          getDescription(), color, getArea(), getPerimeter());
    }
    public String getColor()  { return color; }
    public boolean isFilled() { return filled; }
}
public class Circle extends Shape {
    private double radius;
    public Circle(String color, boolean filled, double radius) {
        super(color, filled);
        if (radius <= 0) throw new IllegalArgumentException("Radius must be positive");
        this.radius = radius;
    }
    @Override public double getArea()        { return Math.PI * radius * radius; }
    @Override public double getPerimeter()   { return 2 * Math.PI * radius; }
    @Override public String getDescription() { return "Circle(r=" + radius + ")"; }
}
public class Rectangle extends Shape {
    private double width, height;
    public Rectangle(String color, boolean filled, double width, double height) {
        super(color, filled);
        this.width = width; this.height = height;
    }
    @Override public double getArea()        { return width * height; }
    @Override public double getPerimeter()   { return 2 * (width + height); }
    @Override public String getDescription() { return "Rectangle(" + width + "x" + height + "
)"; }
}
```

■ DIKKAT

abstract siniftan new Shape() diyerek nesne olusturulamaz! Abstract sinif yalnizca kalitim icin vardir, dogrudan orneklenemez.

12.2 Exception Handling

```java
// try-catch-finally structure
try {
    int[] arr = {1, 2, 3};
    System.out.println(arr[10]);        // throws ArrayIndexOutOfBoundsException
    int result = 10 / 0;                // throws ArithmeticException
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Array error: " + e.getMessage());
} catch (ArithmeticException e) {
    System.out.println("Math error: " + e.getMessage());
} catch (Exception e) {
    // Catches ANY exception not caught above
    System.out.println("Unexpected: " + e.getMessage());
    e.printStackTrace();  // prints full stack trace for debugging
} finally {
    // ALWAYS runs — even if exception occurs or no exception
    System.out.println("This always executes!");
}
// Multi-catch (Java 7+)
try {
    // risky code
} catch (ArrayIndexOutOfBoundsException | NullPointerException e) {
    System.out.println("Array or Null error: " + e.getMessage());
}
```

12.3 Custom Exceptions

```java
// Checked Exception — must be caught or declared with 'throws'
public class InsufficientFundsException extends Exception {
    private double available;
    private double requested;
    public InsufficientFundsException(double available, double requested) {
        super(String.format("Need $%.2f but only $%.2f available",
                            requested, available));
        this.available = available;
        this.requested = requested;
    }
    public double getShortfall() { return requested - available; }
}
// Unchecked Exception — no forced handling
public class InvalidAgeException extends RuntimeException {
    public InvalidAgeException(int age) {
        super("Age " + age + " is not valid (must be 0-150)");
    }
}
// Usage
public void withdraw(double amount) throws InsufficientFundsException {
    if (amount > balance) {
        throw new InsufficientFundsException(balance, amount);
    }
    balance -= amount;
}
// Caller must handle it
try {
    account.withdraw(5000);
```

```
} catch (InsufficientFundsException e) {
    System.out.println(e.getMessage());
    System.out.printf("You are short: $%.2f%n", e.getShortfall());
}
```

Exception Type	Aciklama	Ornek
Checked	Compile-time'da yakalanmak ZORUNDA	IOException, SQLException
Unchecked	RuntimeException — zorunlu degil	NullPointerException, ArrayIndexOut...
Error	JVM-level, yakalanmamali	OutOfMemoryError, StackOverflowError

■ PROJECT: Geometry Calculator + Exception Safe [Intermediate]

➤ Shape (abstract), Circle, Rectangle, Triangle, Square

➤ Throw NegativeDimensionException for invalid inputs

➤ try-catch around all user inputs — never let program crash

➤ Find largest shape by area in a Shape array

➤ Calculate total area of all shapes

➤ BONUS: Serialize shape info to a text format and print

DAY 13 **Collections: ArrayList & HashMap** *~4 hrs*

■ **Gunun Hedefi:** Dinamik veri yapilari olan ArrayList ve HashMap'i ogrenip dizi kisitlamalarindan kurtulmak.

13.1 Array vs ArrayList

Ozellik	Array	ArrayList
Boyut	Sabit — sonradan degismez	Dinamik — otomatik buyur
Tipler	Primitive + nesne	Sadece nesne (wrapper!!)
Metotlar	.length, Arrays.sort()	.add() .remove() .size() .contains() ...
Performans	Biraz daha hizli	Cok az yava — pratik fark yok
Ne zaman?	Boyut bas■ndan belliyse	Boyut bilinmiyorsa / degisiyorsa

13.2 ArrayList in Detail

```Java
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
// Create — use diamond operator <>
ArrayList<String> names = new ArrayList<>();
ArrayList<Integer> scores = new ArrayList<>();  // NOT int!
// Add elements
names.add("Alice");
names.add("Bob");
names.add("Charlie");
names.add(1, "Zara");  // insert at index 1
// Access
System.out.println(names.size());         // 4
System.out.println(names.get(0));         // Alice
System.out.println(names.indexOf("Bob")); // 2
System.out.println(names.contains("Zara")); // true
// Modify
names.set(0, "Anna");   // replace at index 0
// Remove
names.remove("Bob");    // remove by value
names.remove(0);         // remove by index
// Sort
Collections.sort(names);           // alphabetical
Collections.sort(names, (a, b) -> b.compareTo(a));  // reverse
Collections.reverse(names);        // flip order
Collections.shuffle(names);        // randomize
// Convert
List<String> immutable = List.of("A", "B", "C");  // Java 9+
ArrayList<String> mutable = new ArrayList<>(immutable);
// Iterate
for (String name : names) System.out.println(name);
names.forEach(name -> System.out.println(name));  // lambda
names.forEach(System.out::println);               // method reference
```

■ **DIKKAT**

ArrayList yazamazsiniz! Primitive tipler icin wrapper siniflari kullanin: int -> Integer, double -> Double, char -> Character, boolean -> Boolean.

13.3 HashMap in Detail

```Java
import java.util.HashMap;
import java.util.Map;
// HashMap<KeyType, ValueType>
HashMap<String, Integer> grades = new HashMap<>();
// Add entries
grades.put("Alice", 92);
grades.put("Bob",   78);
grades.put("Charlie", 88);
grades.put("Alice", 95);  // OVERWRITES previous value!
// Access
System.out.println(grades.get("Alice"));            // 95
System.out.println(grades.getOrDefault("Dan", 0)); // 0 (safe get)
System.out.println(grades.containsKey("Bob"));     // true
System.out.println(grades.containsValue(88));        // true
System.out.println(grades.size());                   // 3
// Remove
grades.remove("Bob");
// Iterate all entries
for (Map.Entry<String, Integer> entry : grades.entrySet()) {
    System.out.println(entry.getKey() + " -> " + entry.getValue());
}
// Iterate keys only / values only
for (String key : grades.keySet()) System.out.println(key);
for (int val : grades.values())    System.out.println(val);
// putIfAbsent — only adds if key doesn't exist
grades.putIfAbsent("Eve", 85);
// compute — update existing value
grades.compute("Alice", (key, val) -> val + 5);  // Alice: 100
// Word frequency counter — classic HashMap use case
String text = "the quick brown fox jumps over the lazy dog the fox";
HashMap<String, Integer> freq = new HashMap<>();
for (String word : text.split(" ")) {
    freq.put(word, freq.getOrDefault(word, 0) + 1);
}
System.out.println(freq);  // {the=3, fox=2, ...}
```

■ IPUCU

HashMap siralama garantisi vermez! Sirali istiyorsaniz TreeMap (anahtar sirali) veya LinkedHashMap (eklenme sirali) kullanin.

■ PROJECT: Library Management System [Advanced]

➤ Book class: isbn (key), title, author, year, available boolean

➤ HashMap catalog — use ISBN as key

➤ Operations: addBook, removeBook, findByISBN, findByAuthor

➤ Borrow system: ArrayList borrowedList, borrowBook, returnBook

➤ Statistics: total books, books by author, books published after year X

➤ BONUS: Sort books by title, author, year (3 different sorts)

➤ BONUS: Generate a simple library report (print formatted table)

DAY 14 WEEK 2 PROJECT — OOP Application ~6 hrs

■ **Gunun Hedefi:** OOP prensiplerinin tamamini kullanan buyuk bir proje gelistirmek.

Week 2 Summary

OOP Principle	Java Keyword	Purpose
Encapsulation	private, getter/setter	Veri gizleme ve guvenlik
Inheritance	extends, super, @Override	Kod tekrarini onleme
Polymorphism	upcast, downcast, instanceof	Ayni arayuz, farkli davranis
Abstraction	abstract, interface, default	Karmasikligi gizleme
Collections	ArrayList, HashMap, Collections	Dinamik veri saklama

■ PROJECT: BIG PROJECT: Hospital Management System [Advanced]

➤ -- BASE CLASSES --

➤ Person (abstract): name, lastName, tcId, birthDate, contact info

➤ Patient (extends Person): diagnosis, bloodType, insurance, medHistory

➤ Doctor (extends Person): specialization, workDays[], fee, rating

➤ Appointment: patient + doctor + date + time + status (PENDING/DONE/CANCELLED)

➤ -- DATA LAYER --

➤ HospitalSystem: HashMap + HashMap

➤ ArrayList appointments

➤ -- OPERATIONS --

➤ Register patient, add doctor, schedule appointment, cancel appointment

➤ List all doctors by specialization, find available slots

➤ Patient history: all appointments for a patient

➤ -- EXCEPTIONS --

➤ ConflictingAppointmentException, DoctorUnavailableException

➤ -- REPORTS --

➤ Busiest doctor (most appointments), daily revenue, today's schedule

➤ BONUS: Implement Comparable on Doctor to sort by rating

➤ BONUS: Add Nurse class — multiple inheritance via interfaces

■ ***Gelistirme Adimi:*** *Once UML diyagrami ciz: siniflar ve iliskiler. Sonra adim adim siniflar olustur, her adimda test et.*

■ ***Hafta 2 Tamamlandi!*** *OOP'u gercek dunya problemlerine uygulayabiliyorsun. Hafta 3'te Java'nin modern ve ileri konularini ogreneceksin.*

WEEK 3 | ADVANCED JAVA & REAL WORLD

Generics · Lambda · Streams · File I/O · Database · Maven · Clean Code

DAY 15 Generics & Lambda Expressions *~4 hrs*

■ **Gunun Hedefi:** Tip guvenli generic siniflar yazmak ve Lambda ile fonksiyonel programlama yapmak.

15.1 Why Generics?

```Java
// WITHOUT Generics — unsafe, requires casting
ArrayList rawList = new ArrayList();
rawList.add("Hello");
rawList.add(42);             // Mixed types — dangerous!
String s = (String) rawList.get(1);  // ClassCastException at runtime!
// WITH Generics — type-safe at compile time
ArrayList<String> stringList = new ArrayList<>();
stringList.add("Hello");
// stringList.add(42);   // COMPILE ERROR — caught early!
String s2 = stringList.get(0);  // No cast needed — clean!
```

15.2 Generic Classes

```Java
// T = Type parameter (convention: T=Type, E=Element, K=Key, V=Value)
public class Box<T> {
    private T content;
    public Box(T content) { this.content = content; }
    public T getContent()  { return content; }
    public void setContent(T content) { this.content = content; }
    public boolean isEmpty() { return content == null; }
    @Override
    public String toString() { return "Box[" + content + "]"; }
}
// Multiple type parameters
public class Pair<K, V> {
    private K key;
    private V value;
    public Pair(K key, V value) { this.key = key; this.value = value; }
    public K getKey()   { return key; }
    public V getValue() { return value; }
    @Override
    public String toString() { return "(" + key + ", " + value + ")"; }
}
// Usage
Box<String>  nameBox  = new Box<>("Alice");
Box<Integer> scoreBox = new Box<>(95);
Pair<String, Integer> entry = new Pair<>("Alice", 95);
System.out.println(nameBox);   // Box[Alice]
System.out.println(entry);     // (Alice, 95)
```

15.3 Bounded Type Parameters

```Java
// <T extends Comparable<T>> — T must implement Comparable
```

```
public static <T extends Comparable<T>> T max(T a, T b) {
    return (a.compareTo(b) > 0) ? a : b;
}
System.out.println(max(10, 20));          // 20
System.out.println(max("apple", "zebra"));  // zebra
System.out.println(max(3.14, 2.71));      // 3.14
```

15.4 Lambda Expressions

Java

```
// Lambda = anonymous function: (parameters) -> { body }
// Old way: anonymous class
Runnable r1 = new Runnable() {
    @Override public void run() { System.out.println("Running!"); }
};
// Lambda way
Runnable r2 = () -> System.out.println("Running!");
// Single-line lambda (no braces needed)
Runnable r3 = () -> System.out.println("Running!");
// With parameter
java.util.function.Consumer<String> printer = s -> System.out.println(s);
printer.accept("Hello Lambda!");
// With return
java.util.function.Function<Integer, Integer> square = x -> x * x;
System.out.println(square.apply(5));  // 25
// Sorting with lambda
ArrayList<String> names = new ArrayList<>(List.of("Charlie","Alice","Bob"));
names.sort((a, b) -> a.compareTo(b));         // alphabetical
names.sort((a, b) -> b.compareTo(a));         // reverse
names.sort((a, b) -> a.length() - b.length()); // by length
// Method reference — even shorter than lambda
names.sort(String::compareTo);                // same as (a,b) -> a.compareTo(b)
names.forEach(System.out::println);           // same as s -> System.out.println(s)
// Sort custom objects
ArrayList<Person> people = getPeople();
people.sort((p1, p2) -> p1.getName().compareTo(p2.getName()));       // by name
people.sort((p1, p2) -> Integer.compare(p1.getAge(), p2.getAge())); // by age
// Chained sort: name first, then age
people.sort(java.util.Comparator.comparing(Person::getName)
                                .thenComparingInt(Person::getAge));
```

■ PROJECT: Generic Data Structures [Advanced]

- ➤ Generic Stack: push, pop, peek, isEmpty, size, toString
- ➤ Generic Queue: enqueue, dequeue, front, isEmpty, size
- ➤ Generic MinMax>: find min and max
- ➤ Use lambda to sort ArrayList by 3 different criteria
- ➤ Implement Comparator chain: sort by lastName, then firstName
- ➤ BONUS: Generic BinaryTree> with insert + inorder traversal

DAY 16 **Stream API & Functional Programming** *~4 hrs*

■ **Gunun Hedefi:** Stream API ile veri islemek, filter/map/collect operasyonlarini ve Optional sinifini ogrenin.

16.1 Stream API Nedir?

Stream, koleksiyon verilerini deklaratif olarak (ne yapacagini soyle, nasil degil) islemenizi saglar. Donguler yerine zincirleme operasyonlar kullanilir.

Java

```
import java.util.stream.*;
import java.util.List;
import java.util.Optional;
import java.util.Map;
List<Integer> numbers = List.of(1,2,3,4,5,6,7,8,9,10);
// OLD way (imperative)
List<Integer> result = new ArrayList<>();
for (int n : numbers) {
    if (n % 2 == 0) result.add(n * n);
}
// STREAM way (declarative) — same result, 1 line!
List<Integer> result2 = numbers.stream()
    .filter(n -> n % 2 == 0)         // keep evens:  2,4,6,8,10
    .map(n -> n * n)                  // square them: 4,16,36,64,100
    .collect(Collectors.toList());    // collect to list
System.out.println(result2);  // [4, 16, 36, 64, 100]
```

16.2 Key Stream Operations

Java

```
List<String> names = List.of("Alice","Bob","Charlie","Anna","Dave");
// --- INTERMEDIATE operations (return Stream) ---
// filter — keep elements matching condition
names.stream().filter(n -> n.startsWith("A")).forEach(System.out::println);
// Alice, Anna
// map — transform each element
names.stream().map(String::toUpperCase).forEach(System.out::println);
// sorted — sort elements
names.stream().sorted().forEach(System.out::println);            // alphabetical
names.stream().sorted((a,b)->b.compareTo(a)).forEach(System.out::println); // reverse
// distinct — remove duplicates
List.of(1,1,2,2,3).stream().distinct().forEach(System.out::println);  // 1,2,3
// limit / skip
names.stream().limit(3).forEach(System.out::println);  // first 3
names.stream().skip(2).forEach(System.out::println);   // skip first 2
// --- TERMINAL operations (end the stream) ---
long count   = names.stream().filter(n -> n.length() > 3).count();  // 3
Optional<String> first = names.stream().filter(n->n.startsWith("A")).findFirst();
boolean anyLong = names.stream().anyMatch(n -> n.length() > 4);  // true
boolean allShort = names.stream().allMatch(n -> n.length() < 10); // true
// Numeric streams
List<Integer> scores = List.of(85, 90, 72, 88, 65, 95);
int    sum     = scores.stream().mapToInt(Integer::intValue).sum();
double average = scores.stream().mapToInt(Integer::intValue).average().getAsDouble();
int    max     = scores.stream().mapToInt(Integer::intValue).max().getAsInt();
System.out.println("Sum:" + sum + " Avg:" + average + " Max:" + max);
```

16.3 Collectors

```java
List<String> names = List.of("Alice","Bob","Charlie","Anna","Dave");
// Collect to different types
List<String>    nameList  = names.stream().sorted().collect(Collectors.toList());
Set<String>     nameSet   = names.stream().collect(Collectors.toSet());
String          joined    = names.stream().collect(Collectors.joining(", "));
// joined = "Alice, Bob, Charlie, Anna, Dave"
// Group by first letter
Map<Character, List<String>> grouped = names.stream()
    .collect(Collectors.groupingBy(n -> n.charAt(0)));
// {A=[Alice, Anna], B=[Bob], C=[Charlie], D=[Dave]}
// Partition into two groups
List<Integer> scores = List.of(85, 90, 72, 88, 65, 95, 55);
Map<Boolean, List<Integer>> partitioned = scores.stream()
    .collect(Collectors.partitioningBy(s -> s >= 70));
// {true=[85,90,72,88,95], false=[65,55]}
// Average by group
Map<String, Double> avgByDept = employees.stream()
    .collect(Collectors.groupingBy(Employee::getDepartment,
             Collectors.averagingDouble(Employee::getSalary)));
```

■ PROJECT: E-Commerce Analytics (Stream Focus) [Advanced]

- ➤ Create 50+ sample Product objects (id, name, category, price, stock, rating)
- ➤ Stream operations: filter by category, filter by price range, find top-rated
- ➤ Find most expensive 5 products, cheapest 5 products
- ➤ Calculate total inventory value (price * stock) using stream
- ➤ Group products by category, print count and avg price per category
- ➤ Find products with stock < 5 (low inventory alert)
- ➤ BONUS: Find all products where name contains a search keyword

DAY 17 **File I/O** *~3.5 hrs*

■ **Gunun Hedefi:** Dosyaya yazmak, dosyadan okumak ve modern Files API'yi kullanmak.

17.1 Writing Files

Java

```
import java.nio.file.*;
import java.io.*;
import java.util.List;
// Simple write — overwrites if file exists
Path file = Path.of("output.txt");
Files.writeString(file, "Hello from Java!");
// Write a list of lines
List<String> lines = List.of("Line 1", "Line 2", "Line 3");
Files.write(file, lines);
// APPEND mode — adds to existing content
Files.writeString(file, "\nAppended line", StandardOpenOption.APPEND);
// BufferedWriter — for large files (efficient)
try (BufferedWriter writer = new BufferedWriter(new FileWriter("log.txt", true))) {
    writer.write("[INFO] Application started");
    writer.newLine();
    writer.write("[INFO] Processing data...");
    writer.newLine();
} // auto-closed by try-with-resources
```

■ IPUCU

try-with-resources (try (Resource r = ...)) kullanin! Kaynagi (dosya, stream) finally'de manuel kapatmak yerine otomatik kapatir.

17.2 Reading Files

Java

```
// Read entire file as String
String content = Files.readString(Path.of("output.txt"));
System.out.println(content);
// Read all lines into List
List<String> lines = Files.readAllLines(Path.of("output.txt"));
for (int i = 0; i < lines.size(); i++) {
    System.out.println((i+1) + ": " + lines.get(i));
}
// Stream lines (efficient for large files)
Files.lines(Path.of("large.txt"))
    .filter(line -> line.startsWith("ERROR"))
    .limit(10)
    .forEach(System.out::println);
// BufferedReader — classic, still widely used
try (BufferedReader reader = new BufferedReader(new FileReader("data.txt"))) {
    String line;
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
} catch (IOException e) {
    System.err.println("File error: " + e.getMessage());
}
```

17.3 File & Directory Management

```java
Path p = Path.of("data/reports/summary.txt");
// Check
System.out.println(Files.exists(p));          // file exists?
System.out.println(Files.isDirectory(p));     // is it a directory?
System.out.println(Files.isReadable(p));      // can we read it?
// Create directories (all missing parents too)
Files.createDirectories(p.getParent());  // creates data/reports/
// Copy / Move / Delete
Files.copy(source, target, StandardCopyOption.REPLACE_EXISTING);
Files.move(source, target, StandardCopyOption.REPLACE_EXISTING);
Files.delete(p);              // throws if not found
Files.deleteIfExists(p);      // safe version
// List files in directory
Files.list(Path.of("."))
     .filter(Files::isRegularFile)
     .sorted()
     .forEach(f -> System.out.println(f.getFileName()));
// Walk entire directory tree
Files.walk(Path.of("src"))
     .filter(f -> f.toString().endsWith(".java"))
     .forEach(f -> System.out.println(f));
```

■ DIKKAT

Dosya islemlerinde her zaman IOException yakalanmali ya da throws ile uste iletilmeli. Dosya bulunamama, izin sorunu gibi hatalar cok yaygindir.

■ PROJECT: CSV & Logging System [Intermediate]

- ➤ Save student data to CSV file: name, surname, scores (comma-separated)
- ➤ Read CSV back, create Student objects, parse all fields
- ➤ Calculate average, max, min from file data, write results to output.txt
- ➤ Logger class: log(level, message) writes timestamped lines to log.txt
- ➤ Log levels: INFO, WARN, ERROR — each in different format
- ➤ BONUS: Read a large text file, count word frequencies, write top 20 to file

DAY 18 **Algorithms & Big-O Complexity** *~4 hrs*

■ **Gunun Hedefi:** Temel siralama ve arama algoritmalarini, Big O notasyonunu ve Java'da algoritma yazmay■ ogrenin.

18.1 Big-O Notation

Notation	Name	n=1000	Example
O(1)	Constant	1 op	Array index access, HashMap.get()
O(log n)	Logarithmic	~10 ops	Binary search, TreeMap operations
O(n)	Linear	1,000 ops	Loop through list once
O(n log n)	Linearithmic	~10,000 ops	Arrays.sort(), merge sort
O(n^2)	Quadratic	1,000,000 ops	Bubble sort, nested loops
O(2^n)	Exponential	Huge!	Brute force subset enumeration

18.2 Linear vs Binary Search

Java

```java
// Linear Search — O(n) — checks every element
public static int linearSearch(int[] arr, int target) {
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == target) return i;  // found!
    }
    return -1;  // not found
}
// Binary Search — O(log n) — ARRAY MUST BE SORTED!
public static int binarySearch(int[] arr, int target) {
    int left = 0, right = arr.length - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;  // avoids integer overflow!
        if      (arr[mid] == target) return mid;
        else if (arr[mid] <  target) left = mid + 1;
        else                         right = mid - 1;
    }
    return -1;
}
// Binary search example trace:
// arr = [1,3,5,7,9,11,13,15,17,19], target = 11
// Step 1: mid=9 (arr[9]=19) 11<19 -> right=8
// Step 2: mid=4 (arr[4]=9)  11>9  -> left=5
// Step 3: mid=6 (arr[6]=13) 11<13 -> right=5
// Step 4: mid=5 (arr[5]=11) FOUND at index 5
```

■ DIKKAT

Binary Search SADECE sirali dizilerde calisir! Sirasiz dizide yanlis sonuc verir. Arrays.binarySearch() de ayni kisitlamaya sahiptir.

18.3 Sorting Algorithms

Java

```
// Bubble Sort — O(n^2) — educational only, NOT for production
public static void bubbleSort(int[] arr) {
    for (int i = 0; i < arr.length - 1; i++) {
        boolean swapped = false;
        for (int j = 0; j < arr.length - 1 - i; j++) {
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j]   = arr[j+1];
                arr[j+1] = temp;
                swapped  = true;
            }
        }
        if (!swapped) break;  // already sorted, early exit
    }
}
// Selection Sort — O(n^2) — finds min in each pass
public static void selectionSort(int[] arr) {
    for (int i = 0; i < arr.length - 1; i++) {
        int minIdx = i;
        for (int j = i+1; j < arr.length; j++) {
            if (arr[j] < arr[minIdx]) minIdx = j;
        }
        int temp = arr[i]; arr[i] = arr[minIdx]; arr[minIdx] = temp;
    }
}
// In production: ALWAYS use Arrays.sort() — O(n log n) Timsort
int[] data = {5, 2, 8, 1, 9};
Arrays.sort(data);  // fast, tested, optimized
```

■ IPUCU

Production kodunda asla manuel sort yazmayiniz. Arrays.sort() ve Collections.sort() optimize edilmis Timsort kullanir. Manuel sort yalnizca ogrenme amacl■dir.

■ PROJECT: Algorithm Benchmark [Intermediate]

- ➤ Generate arrays of size: 1000, 10000, 100000
- ➤ Time linear vs binary search (use System.nanoTime())
- ➤ Time bubble sort vs Arrays.sort() — compare results
- ➤ Print results in a formatted table
- ➤ Implement recursive binary search, compare with iterative
- ➤ BONUS: Implement merge sort — O(n log n)
- ➤ BONUS: Count comparisons made in each sort algorithm

DAY 19

Database with Java — SQLite / JDBC

~4.5 hrs

■ **Gunun Hedefi:** JDBC ile veritabani baglantisi kurmak, temel SQL sorgularini Java'dan calistirmak.

19.1 JDBC & SQLite Setup

Java

```
// Add to Maven pom.xml:
<dependency>
    <groupId>org.xerial</groupId>
    <artifactId>sqlite-jdbc</artifactId>
    <version>3.45.1.0</version>
</dependency>
// Manual JAR: https://github.com/xerial/sqlite-jdbc/releases
// IntelliJ: File -> Project Structure -> Modules -> Dependencies -> + JAR
```

19.2 Connection & Table Creation

Java

```
import java.sql.*;
public class DatabaseManager {
    private static final String URL = "jdbc:sqlite:school.db";
    public static Connection connect() throws SQLException {
        return DriverManager.getConnection(URL);
    }
    public static void createTable() {
        String sql = """
            CREATE TABLE IF NOT EXISTS students (
                id       INTEGER PRIMARY KEY AUTOINCREMENT,
                name     TEXT    NOT NULL,
                surname  TEXT    NOT NULL,
                gpa      REAL    DEFAULT 0.0,
                email    TEXT    UNIQUE
            )""";
        try (Connection con = connect();
             Statement stmt = con.createStatement()) {
            stmt.execute(sql);
            System.out.println("Table created!");
        } catch (SQLException e) {
            System.err.println("Error: " + e.getMessage());
        }
    }
}
```

19.3 CRUD Operations

Java

```
// CREATE — insert record
public static void addStudent(String name, String surname, double gpa) {
    String sql = "INSERT INTO students (name, surname, gpa) VALUES (?, ?, ?)";
    // PreparedStatement prevents SQL Injection!
    try (Connection con = connect();
         PreparedStatement ps = con.prepareStatement(sql)) {
        ps.setString(1, name);
        ps.setString(2, surname);
        ps.setDouble(3, gpa);
```

```
        int rows = ps.executeUpdate();
        System.out.println(rows + " row(s) inserted.");
    } catch (SQLException e) { e.printStackTrace(); }
}
// READ — query records
public static void listStudents() {
    String sql = "SELECT * FROM students ORDER BY gpa DESC";
    try (Connection con = connect();
         Statement stmt  = con.createStatement();
         ResultSet rs    = stmt.executeQuery(sql)) {
        System.out.printf("%-5s %-15s %-15s %5s%n","ID","Name","Surname","GPA");
        System.out.println("-".repeat(44));
        while (rs.next()) {
            System.out.printf("%-5d %-15s %-15s %.2f%n",
                rs.getInt("id"), rs.getString("name"),
                rs.getString("surname"), rs.getDouble("gpa"));
        }
    } catch (SQLException e) { e.printStackTrace(); }
}
// UPDATE
public static void updateGpa(int id, double newGpa) {
    String sql = "UPDATE students SET gpa = ? WHERE id = ?";
    try (Connection con = connect();
         PreparedStatement ps = con.prepareStatement(sql)) {
        ps.setDouble(1, newGpa);
        ps.setInt(2, id);
        ps.executeUpdate();
    } catch (SQLException e) { e.printStackTrace(); }
}
// DELETE
public static void deleteStudent(int id) {
    String sql = "DELETE FROM students WHERE id = ?";
    try (Connection con = connect();
         PreparedStatement ps = con.prepareStatement(sql)) {
        ps.setInt(1, id);
        ps.executeUpdate();
    } catch (SQLException e) { e.printStackTrace(); }
}
```

■ HATA

String birlestirme ile SQL ASLA yazmayin: "...WHERE name='" + input + "'" SQL Injection saldir■s■na aciktir! Her zaman PreparedStatement kullanin.

■ PROJECT: Library Database App [Advanced]

➤ SQLite: books table — id, isbn, title, author, year, copies

➤ CRUD: addBook, listAll, findByAuthor, findByISBN, updateCopies, delete

➤ BorrowRecord table: id, isbn, borrower_name, borrow_date, return_date

➤ All queries use PreparedStatement (no SQL injection!)

➤ Interactive console menu with full CRUD access

➤ BONUS: JOIN query — show all currently borrowed books with borrower names

DAY 20 **Maven, Build Tools & Clean Code** *~3.5 hrs*

■ **Gunun Hedefi:** Maven ile bagimlilik yonetimini, temiz kod prensiplerini ve temel test yazmay■ ogrenin.

20.1 Maven Basics

Java

```
<!-- pom.xml — project configuration -->
<project xmlns="http://maven.apache.org/POM/4.0.0">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.yourname</groupId>
    <artifactId>my-project</artifactId>
    <version>1.0.0</version>
    <properties>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <!-- Search for libraries at: https://mvnrepository.com -->
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.10.1</version>
        </dependency>
        <!-- JUnit 5 for testing -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.10.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
// Maven commands:
// mvn compile     — compile source code
// mvn test        — run tests
// mvn package     — create JAR
// mvn clean       — delete build files
// mvn clean install — full rebuild
```

20.2 Clean Code Principles

Bad Code	Clean Code	Why?
int d = 86400;	int SECONDS_PER_DAY = 86400;	Magic numbers
void proc()	void saveStudentRecord()	Descriptive names
if (flag == true)	if (flag)	Redundant comparison
catch (Exception e) {}	Catch specific exceptions	Silent failures
200-line method	Max 20-30 lines, single purpose	SRP principle
// increment x	No comment needed, code is clear	Self-documenting code
class DataManager {huge}	Split into Reader, Writer, Validator	SRP principle

20.3 Writing Tests with JUnit 5

```java
import org.junit.jupiter.api.*;
import static org.junit.jupiter.api.Assertions.*;
public class CalculatorTest {
    private Calculator calc;
    @BeforeEach  // runs before each test
    void setUp() {
        calc = new Calculator();
    }
    @Test
    void testAddition() {
        assertEquals(5, calc.add(2, 3));
        assertEquals(0, calc.add(-1, 1));
        assertEquals(-3, calc.add(-1, -2));
    }
    @Test
    void testDivisionByZero() {
        assertThrows(ArithmeticException.class, () -> calc.divide(10, 0));
    }
    @Test
    void testNegativeResult() {
        assertTrue(calc.subtract(3, 5) < 0);
    }
    @Test @DisplayName("Multiplication table")
    void testMultiply() {
        assertAll(
            () -> assertEquals(6,   calc.multiply(2, 3)),
            () -> assertEquals(-6,  calc.multiply(-2, 3)),
            () -> assertEquals(0,   calc.multiply(0, 100))
        );
    }
}
```

■ PROJECT: Maven Project: JSON + Tests [Advanced]

➤ Create a Maven project in IntelliJ IDEA

➤ Add Gson + JUnit5 dependencies to pom.xml

➤ Create Student POJO, serialize to JSON with Gson

➤ Read JSON file, deserialize back to Student objects

➤ Write JUnit5 tests for all methods in a utility class

➤ Achieve 100% test coverage on your utility methods

➤ BONUS: Add SLF4J + Logback for proper logging

DAY 21

FINAL PROJECT — Full Console Application

~8 hrs

■ **Gunun Hedefi:** 21 gunde ogrenilenTUM konulari birlestiren, gercek dunya problemini cozen kapsamli bir uygulama gelistirmek.

21-Day Summary

Week	Topics	Tools
Week 1	Variables, Conditionals, Loops, Arrays, Methods, Strings	Java Core
Week 2	OOP: Classes, Inheritance, Polymorphism, Interfaces, Collections	Design Patterns
Week 3	Generics, Lambda, Streams, I/O, Database, Maven, Clean Code	Modern Java

■ PROJECT: FINAL: Restaurant Management System [Expert]

➤ --- OOP LAYER ---

➤ MenuItem: private id, name, category, price, stock — full encapsulation

➤ Order: tableNumber, ArrayList, timestamp, Status enum

➤ OrderItem: MenuItem + quantity + subtotal

➤ Table: capacity, status (FREE/OCCUPIED), activeOrder

➤ --- DATA LAYER ---

➤ SQLite: menu_items, orders, order_items tables

➤ DAO pattern: MenuDAO, OrderDAO — CRUD for each (PreparedStatement only)

➤ --- BUSINESS LOGIC ---

➤ Place order, add/remove items, process payment, clear table

➤ Stream API: daily revenue report, most ordered items

➤ File I/O: export daily report to .txt file

➤ --- CLEAN CODE ---

➤ Maven project, single-responsibility classes

➤ Custom exceptions: OutOfStockException, TableOccupiedException

➤ Unit tests for MenuItem, Order calculation logic

➤ BONUS: Colorize console output with ANSI escape codes

➤ BONUS: Add a discount system with different types (%, fixed, BOGO)

■ Bundan Sonra Ne Yapmal■?

- **Spring Boot:** Java ile web/REST API gelistirmenin en populer framework'u — en onemli sonraki adim
- **Data Structures & Algorithms:** LeetCode/HackerRank'te pratik yap, mulakatlara hazirlan
- **Git & GitHub:** Version control — her proje icin zorunlu, acik kaynak projelere katki yap
- **JUnit 5 + Mockito:** Profesyonel test yazimi icin sart
- **Design Patterns:** Singleton, Factory, Builder, Observer, Strategy ogren
- **Docker & CI/CD:** Uygulamayi containerize et, GitHub Actions ile deploy et
- **Hibernate / JPA:** ORM ile veritabani islemlerini kolaylastir
- **Microservices:** Spring Cloud ile dagilmis sistemler gelistir

■ Tebrikler! 21 Gunluk Java Bootcamp'i Tamamlad■n.

Surekli kod yaz, gercek projeler gelistir, acik kaynak projelere katki yap. En iyi ogrenme yontemi YAPMAKTIR — ogrenmede devam et!