Files & the FRQ

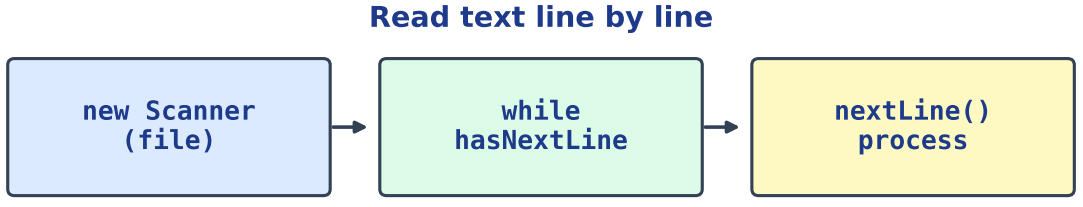
Java Reference

Text files with Scanner

A `Scanner` reads text one line at a time. For a real file you write `new Scanner(new File("scores.txt"))`; here we wrap a `String` so the example runs anywhere. Use `.split(" ")` to split 拆分 a line into parts and `Integer.parseInt(...)` to parse 解析 a number from text.

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Real file:  Scanner in = new Scanner(new File("scores.txt"));
        String data = "Alice 80\nBob 95\nCara 72";
        Scanner in = new Scanner(data);
        int total = 0, count = 0;
        while (in.hasNextLine()) {
            String line = in.nextLine();
            String[] parts = line.split(" ");      // break the line on
 ↪  the space
            total += Integer.parseInt(parts[1]);
            count++;
        }
        System.out.println("average = " + (total / count));   // average
 ↪  = 82
    }
}
```



Exam tip: hasNextLine guards the loop; split + parseInt turn a line into data

Scanner reads lines while hasNextLine is true

The AP FRQ question types

The AP CS A exam has four free-response 自由作答 questions, each a fixed shape:

- **Q1 — Methods & control:** write methods to a given spec; loops, `if`, `String`/`Math`.
- **Q2 — Class design:** write a full class (fields, constructor, methods) from a description.
- **Q3 — Array / ArrayList:** process a 1-D array or `ArrayList` (search, count, build a new list).
- **Q4 — 2-D array:** traverse a grid by row and column.

The skill is always the same: read the spec, write the method exactly as described, return the right type.

```
public class Main {
    public static void main(String[] args) {
        // Q1 style: implement a method to a spec, then it is tested.
        System.out.println(countEven(new int[]{4, 7, 10, 3, 6}));   // 3
    }

    /** Returns how many values in arr are even. */
    public static int countEven(int[] arr) {
        int count = 0;
        for (int x : arr)
            if (x % 2 == 0) count++;
        return count;
    }
}
```

Common mistakes

- `nextInt()` leaves the newline behind, so a following `nextLine()` reads an empty line — read it away first.
- Check `hasNext()` before reading, to avoid running off the end of the file.
- In the FRQ, read the method header carefully: match the return type and parameters exactly.