Variables in Java

1. Introduction to Variables in Java

In Java, variables are fundamental building blocks used to store data that can be accessed and manipulated throughout a program. Variables in Java are containers that hold data values, and each variable has a specific data type that defines the type of value it can store. These data types range from primitive types, such as integers, floating-point numbers, characters, and booleans, to complex objects, which are created from classes. Java enforces strict type-checking, meaning each variable must be declared with a specific data type before being used, ensuring data integrity and reducing runtime errors.

Key Points:

- **Declaration**: Every variable must be declared with a data type before use.
- **Initialization**: Variables can be initialized at the time of declaration.
- **Scope**: Variables in Java have a scope, determining where in the code they can be accessed.
- **Lifetime**: A variable's lifetime is determined by where it is declared, and it exists only within that scope.

2. Example Code for Variables in Java

Here is an example in Java demonstrating how to declare and initialize different types of variables.

```java
public class VariableExamples {
    public static void main(String[] args) {
        // Numeric variables
        int age = 25; // Integer type
        double salary = 45000.75; // Double type

        // Text variable
        String name = "Alice"; // String type for text

        // Boolean variable
        boolean isStudent = true; // Boolean type

        // Print the variables
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Salary: " + salary);
        System.out.println("Is Student: " + isStudent);
    }
}
```

3. Data Types in Java (JDK 23)

The table below provides a comprehensive overview of data types available in JDK 23, organized by category. Each data type has specific limits and occupies a set amount of memory, as shown in the table.

Category	Data Type	Valid Limits	Memory Size	Example Declaration in Java
Numeric	`byte`	-128 to 127	1 byte	`byte b = 100;`
	`short`	-32,768 to 32,767	2 bytes	`short s = 20000;`
	`int`	-2,147,483,648 to 2,147,483,647	4 bytes	`int i = 500000;`
	`long`	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	8 bytes	`long l = 5000000000L;`
	`float`	Approximately ±3.4e−038 to ±3.4e+038	4 bytes	`float f = 3.14f;`
	`double`	Approximately ±1.7e−308 to ±1.7e+308	8 bytes	`double d = 3.14159;`
Text	`char`	Single Unicode character	2 bytes	`char c = 'A';`
	`String`	Sequence of characters	Varies (object)	`String s = "Hello World";`
Boolean	`boolean`	`true` or `false`	1 bit	`boolean flag = true;`

4. Invalid Variable Assignments

In Java, each data type has specific requirements and limitations. Here are some common invalid variable assignments that would cause errors in Java:

- **Invalid `float` assignment (missing 'f' suffix)**:

  ```
  float f = -35.0; // Error: floating-point literal requires an 'f' suffix for float type
  ```

  Correction:

  ```
  float f = -35.0f;
  ```

- **Out-of-bounds `byte` assignment**:

  ```
  byte b = 150; // Error: value exceeds the range of byte (-128 to 127)
  ```

  Correction:

  ```
  byte b = 127;
  ```

- **Out-of-bounds `short` assignment**:

  ```
  short s = 40000; // Error: value exceeds the range of short (-32,768 to 32,767)
  ```

  Correction:

  ```
  short s = 32767;
  ```

- **Incorrect `char` assignment (using multiple characters)**:

  ```
  char c = 'AB'; // Error: char can only hold a single character
  ```

  Correction:

  ```
  char c = 'A';
  ```

- **Invalid `boolean` assignment (using numeric value)**:

  ```
  boolean flag = 1; // Error: boolean can only be true or false
  ```

  Correction:

  ```
  boolean flag = true;
  ```

5. Exercises Using Different Types of Variables

1. **Exercise 1**: Declare an `int` variable named `year` and set it to the current year. Print the variable's value.

2. **Exercise 2**: Declare a `double` variable called `temperature` and set it to a value with one decimal point. Print out the temperature.

3. **Exercise 3**: Create a `String` variable called `city` and set it to the name of your hometown. Display the city name.

4. **Exercise 4**: Define a `boolean` variable named `isAvailable` and set it to `false`. Print out the availability status.

5. **Exercise 5**: Declare a `char` variable named `grade` and set it to any letter between 'A' and 'F'. Print the grade.