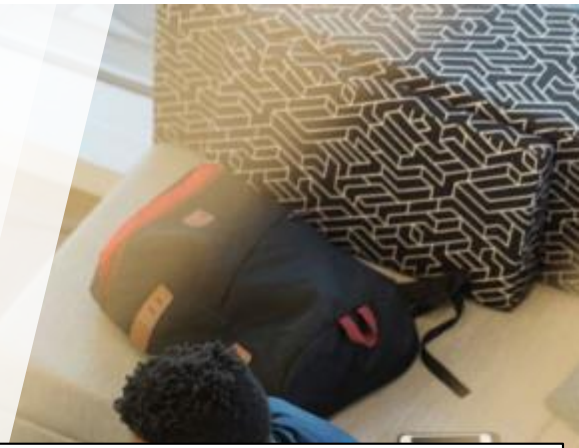
Java Basics

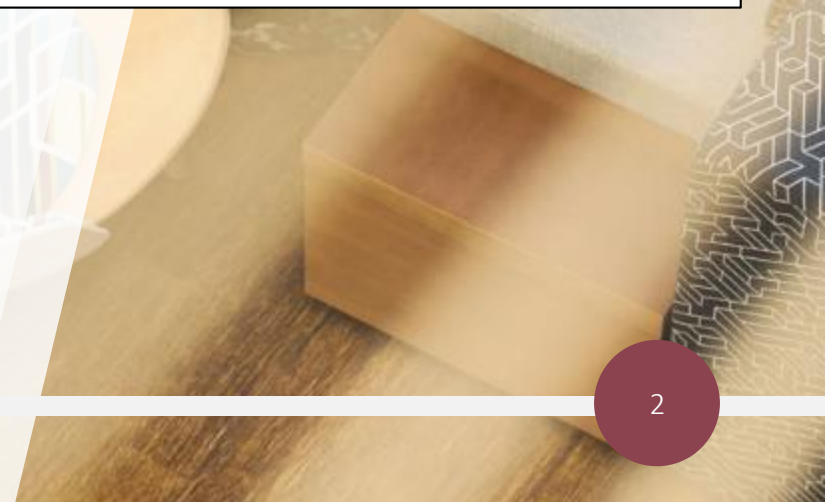
Java Basics



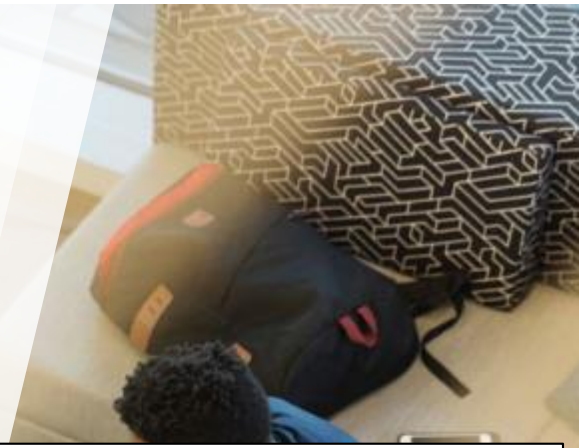
Java 8 OCA (1Z0-808)

Java Basics

- ✓ Define the scope of variables
- ✓ Define the structure of a Java class
- ✓ Create executable Java applications with a main method; run a Java program from the command line; produce console output
- ✓ Import other Java packages to make them accessible in your code
- ✓ Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation, etc.



Java Basics



Java 8 OCA (1Z0-808)

Java Basics

- Define the scope of variables
- Define the structure of a Java class
- Create executable Java applications with a main method; run a Java program from the command line; produce console output
- Import other Java packages to make them accessible in your code
- Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation, etc.

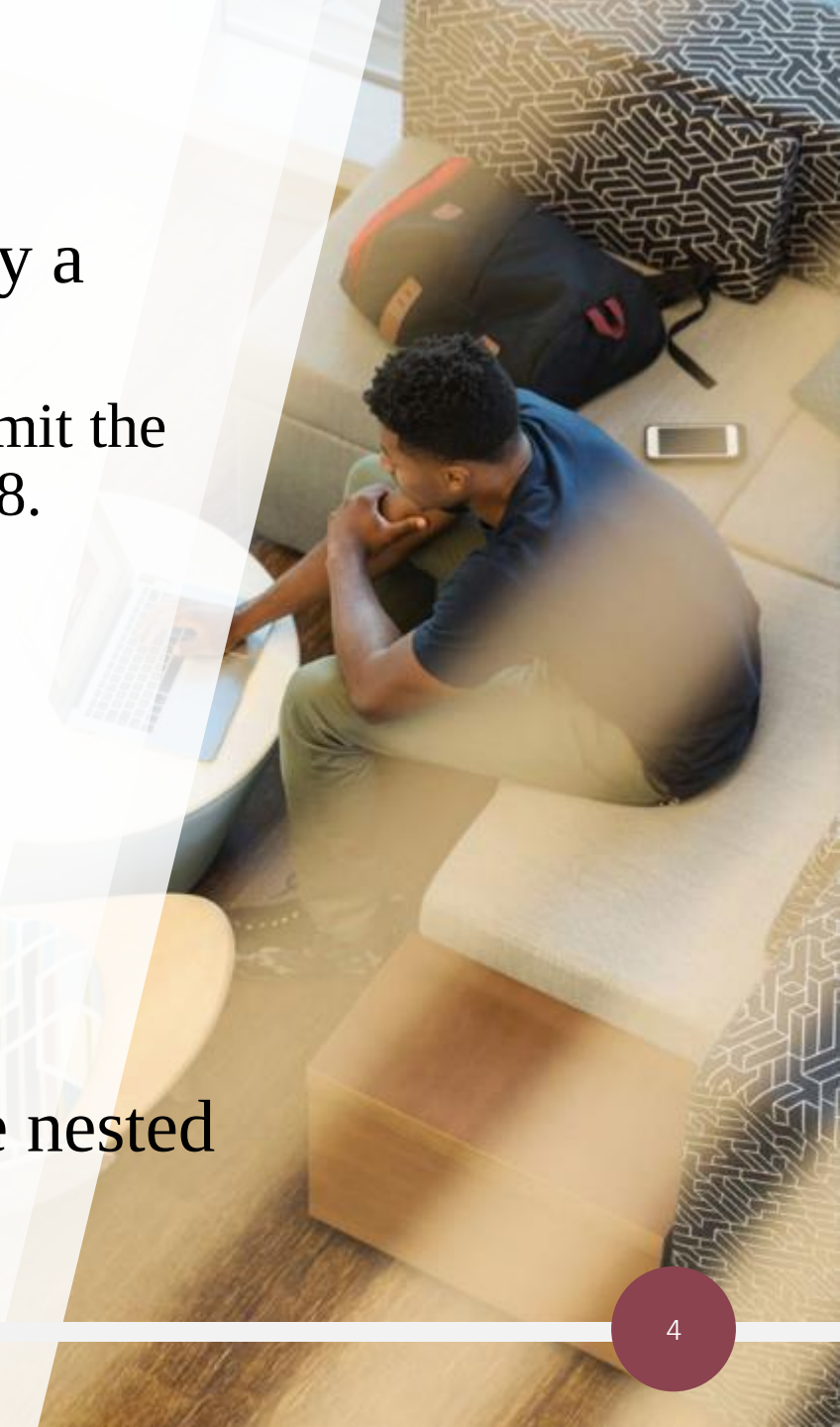
Scope

- Java is a strongly-typed language i.e. you must specify a variable's type when declaring it.
  - ➢ in Java 10, Local Variable Type Inference enables you to omit the type of the local variable. Obviously not applicable to Java 8.

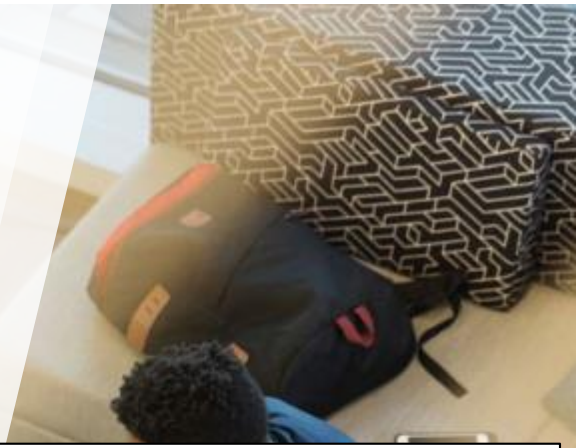
- Java uses "block" scope.

- A block is created with a set of braces i.e. { }.

- Blocks can be nested and the scope is valid within the nested blocks.



Java Basics

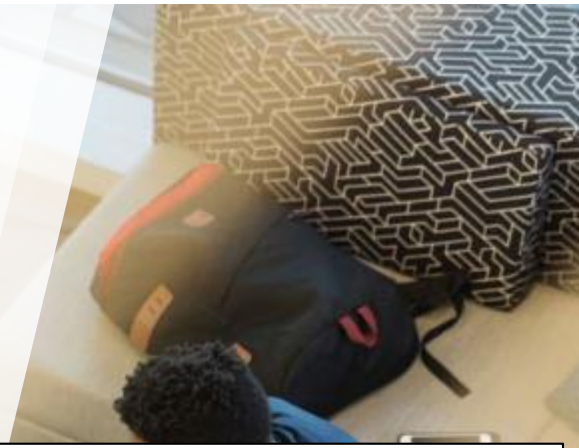


Java 8 OCA (1Z0-808)

Java Basics

- ✓ Define the scope of variables
- ✓ Define the structure of a Java class
- Create executable Java applications with a main method; run a Java program from the command line; produce console output
- Import other Java packages to make them accessible in your code
- ✓ Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation, etc.

Java Basics



Java 8 OCA (1Z0-808)

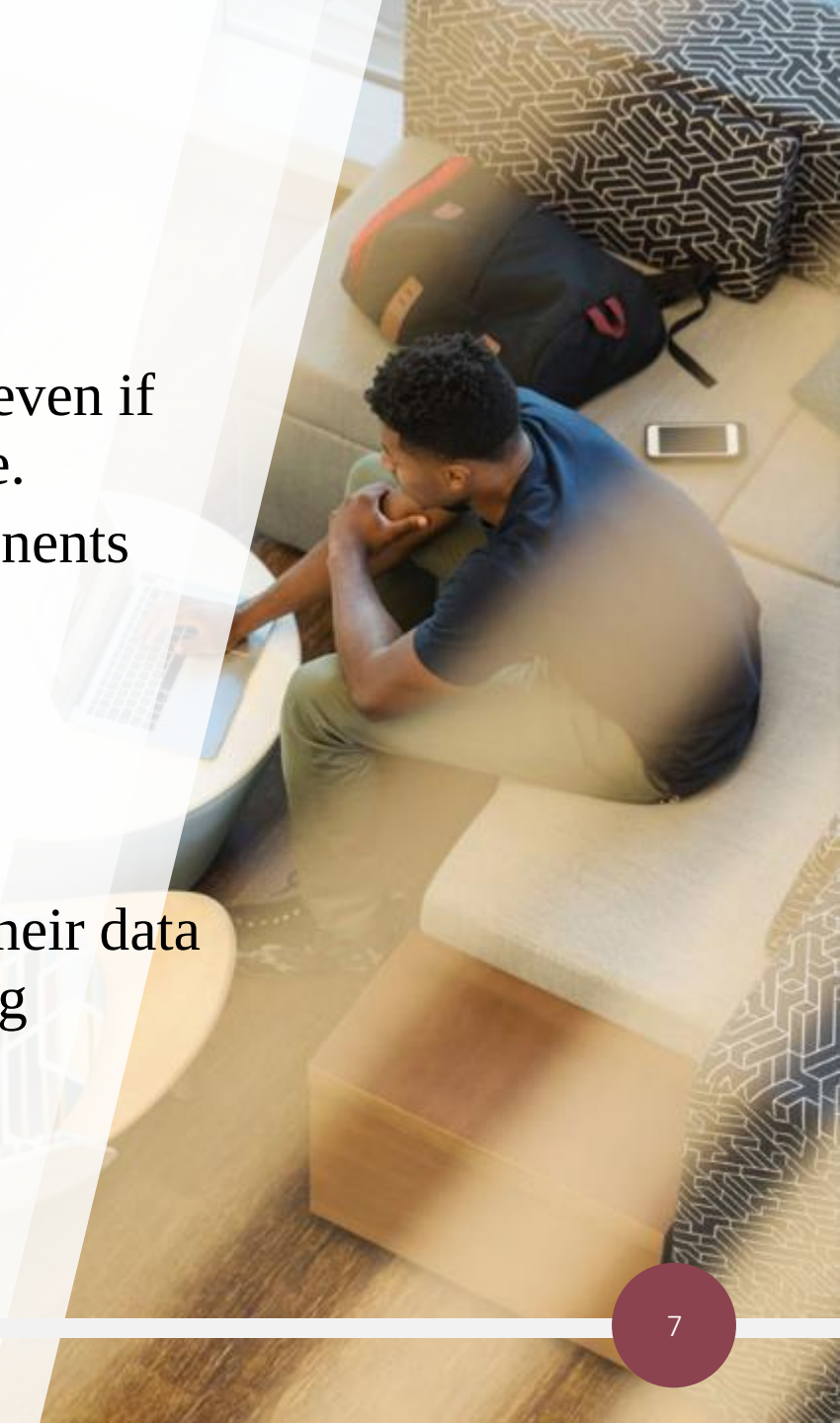
Java Basics

- ✓ Define the scope of variables
- ✓ Define the structure of a Java class
- ✓ Create executable Java applications with a main method; run a Java program from the command line; produce console output
- ✓ Import other Java packages to make them accessible in your code
- → Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation, etc.

Features and Components of Java

- Object oriented
  - Well designed OO systems remain testable and enhanceable even if they grow into huge applications with millions of lines of code.
  - OO design offers a natural way to think about how the components interact.

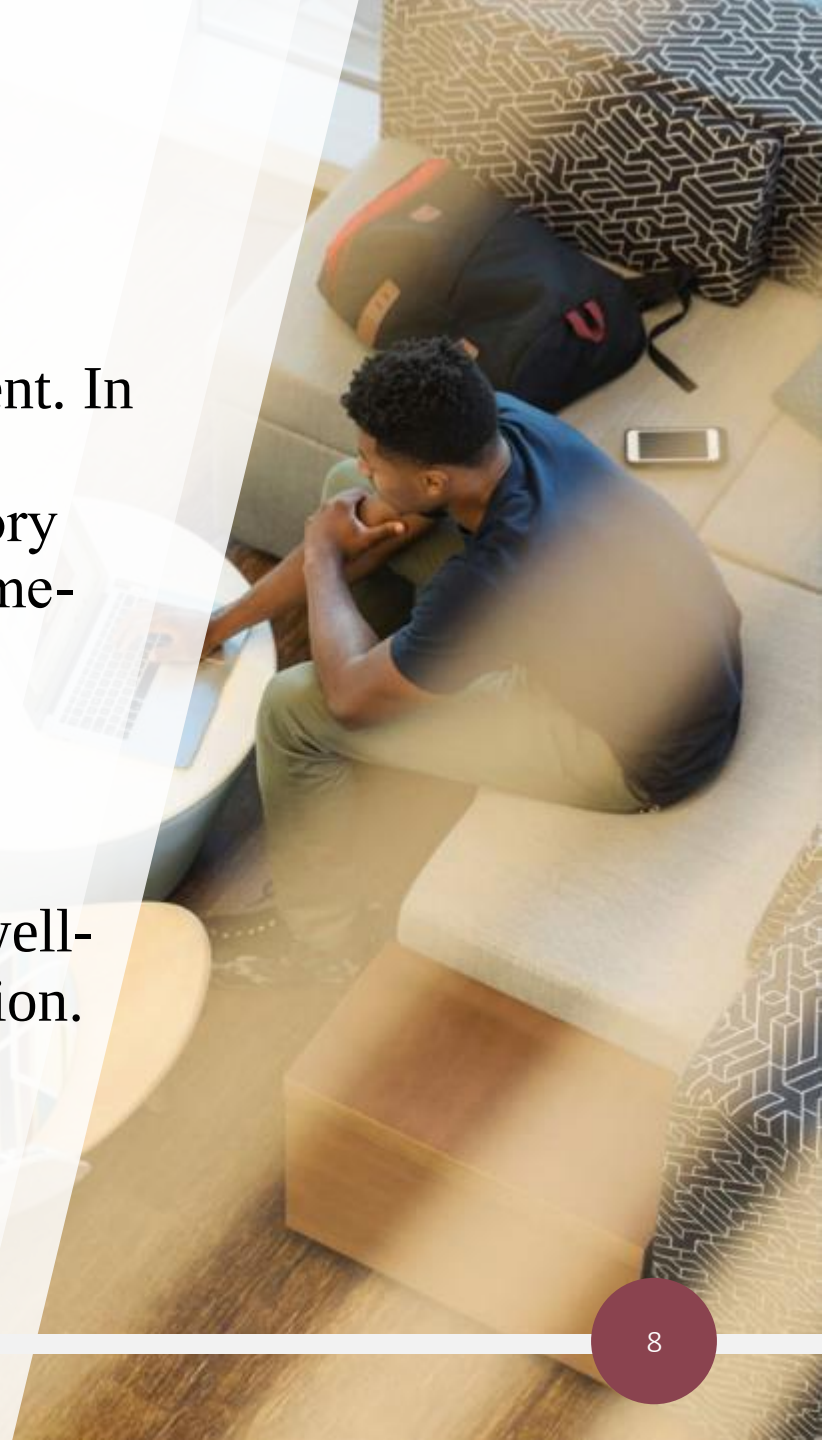
- Encapsulation
  - A core pillar of OOP enabling software components to hide their data from other components; thereby, protecting the data from being updated without the components approval or knowledge.



Features and Components of Java

- Memory management
  - Unlike C or C++, Java provides automatic memory management. In languages that do not provide automatic memory management, keeping track of memory usage is complex and leads to “memory leaks”. Tracking down and fixing memory leaks is common, time-consuming and error-prone.

- Huge library
  - Java has an enormous library of pre-written, well-tested and well-documented code. This code is easy to include in your application.

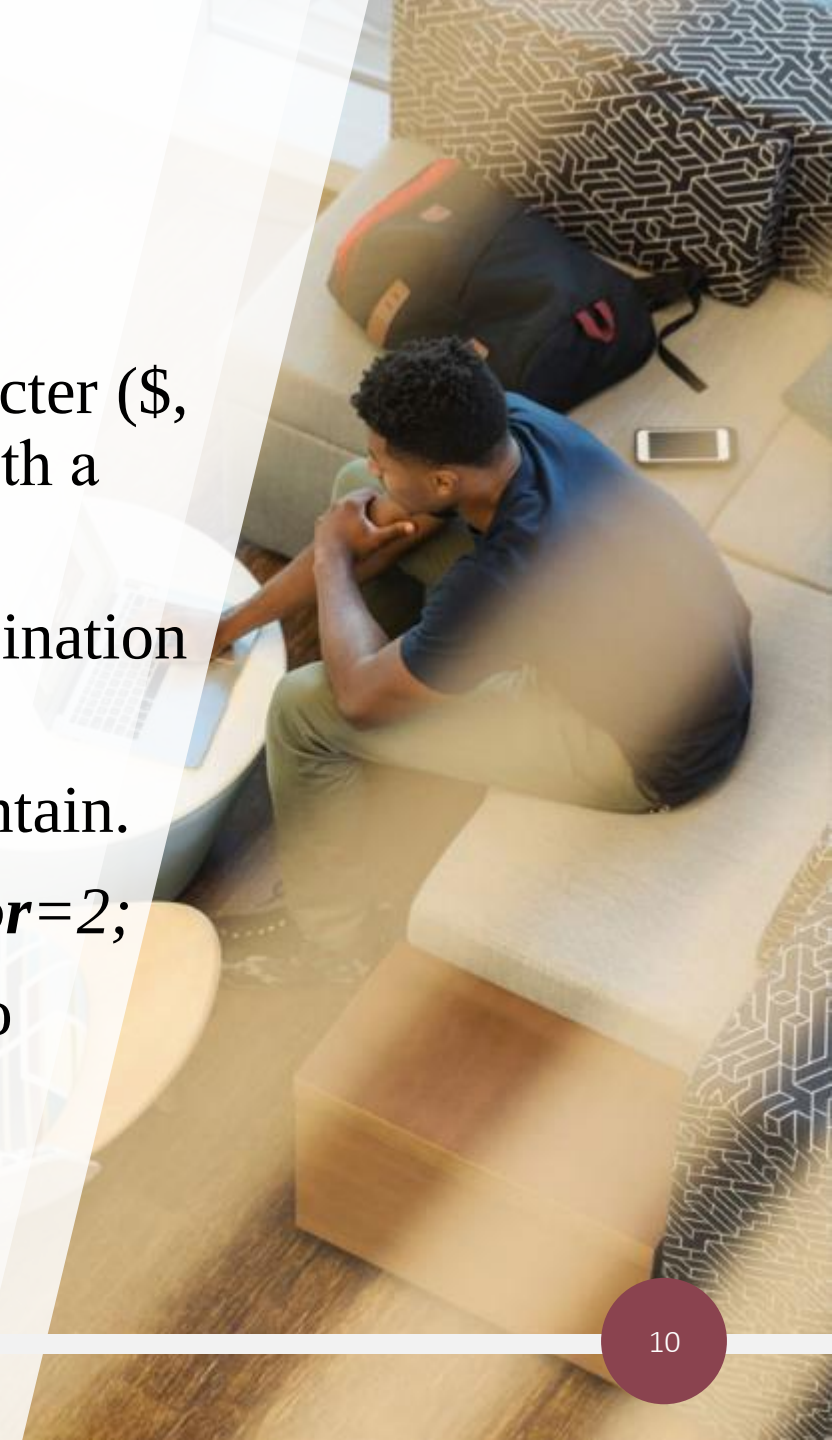


Features and Components of Java

- Platform Independence
  - Java code can be written on one platform and by using a platform-specific JVM, can execute on that platform. This is known as “write once, run anywhere”.

Java Basics

- Legal identifiers
  - Legal identifiers must start with a letter, a currency character ($, £, €), or an underscore (_). Identifiers CANNOT begin with a digit!.
  - After the first character, identifiers can contain any combination of letters, currency characters, underscores or numbers.
  - No limit to the number of characters an identifier can contain.
  - A Java keyword cannot be used as an identifier e.g. *int* ***for****=2;*
  - Java identifiers ARE case sensitive; *foo* and FOO are two different identifiers.



Java Basics

- Legal identifiers examples:
  - int _a; int €6; int £3; int $h_8; int _; int $c; int ____c_w; int _$;
  - int this_is_a_very_long_name_for_an_identifier;

- Illegal identifiers examples:
  - int :b;
  - int –d;
  - int e#; // # not allowed
  - int .f;
  - int 7g;

Java Language Keywords

Here is a list of keywords in the Java programming language. You cannot use any of the following as identifiers in your programs. The keywords `const` and `goto` are reserved, even though they are not currently used. `true`, `false`, and `null` might seem like keywords, but they are actually literals; you cannot use them as identifiers in your programs.

`abstract`	`continue`	`for`	`new`	`switch`
`assert`***	`default`	`goto`*	`package`	`synchronized`
`boolean`	`do`	`if`	`private`	`this`
`break`	`double`	`implements`	`protected`	`throw`
`byte`	`else`	`import`	`public`	`throws`
`case`	`enum`****	`instanceof`	`return`	`transient`
`catch`	`extends`	`int`	`short`	`try`
`char`	`final`	`interface`	`static`	`void`
`class`	`finally`	`long`	`strictfp`**	`volatile`
`const`*	`float`	`native`	`super`	`while`

* not used

** added in 1.2

*** added in 1.4

**** added in 5.0

docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html