Module 3: Iteration & Classes

Loop Constructs

Main Components of a Loop

Initialisation of a counter or flag - The counter or flag initialises once and tracks the execution of the loop.

Boolean condition - stop or exit the loop when the condition is met.

Increment/Decrement or reset - increase or decrease the counter or flag state.

Pre-Test Loops - For Loop

```
public static void main(String[] args) {
        // for ( Initialization ; Boolean Condition ; Increment/Decrement)
        for (int i = 0; i < max; i++) {
              // Instructions
         }
}
```

Pre-Test Loops - While Loop

```
public static void main(String[] args) {
        // Initialization
        int max = 10;
        int i = 0;

        // While Loop with Boolean Condition
        while ( i < max) {
              // Instructions
              // Increment/Decrement
              i++;
         }
}
```

Post-Test Loops - Do-While Loop

```
public static void main(String[] args) {
        // Initialization
        int max = 10;
        int i = 0;

        do {
              // Instructions
              // Increment/Decrement
              i++;
        } while (x < max); // Boolean Condition gets checked here
}
```

Break	A statement used to terminate the execution of a loop.
Continue	A statement used to skip the current iteration of the loop.

Collections { Basic Data Structures }

Computers require a way to store pieces of data. Data structures organises data that it facilitates easy traversal across the multiple pieces of data. Java.util.collections framework support three main types - linked lists, queues and sets.

Arrays

Very fast but has to be instantiated with a predefined size due to Computing systems allocates memory for an entire array of elements sequentially.

10	18	13	20	12
a[0]	a[1]	a[2]	a[3]	a[4]

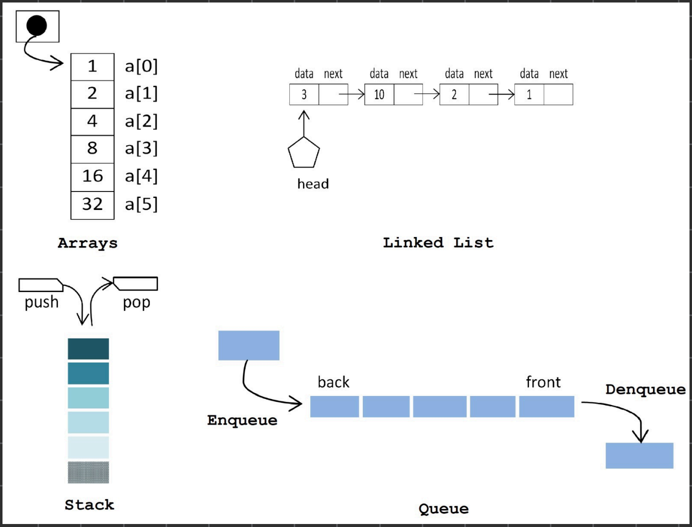
< Datatype >[] < Array Name> = new < Datatype > [Length]

Declare Declare the array and specify the name and primitive or object datatype.

Length In Jawa, arrays are fixed in length and are not dynamic. It has to be defined at the creation of the away for java to sequentially locate these data to memory.

Initialise

1) Best practice is to have objects explicit initialised. Else objects und strings will have a "Null" value assigned by default if not initialised.
2) This may cause "Null Pointer Exception" error. Java will try to reference an object but gets a "Null" value which will crash the execution.
3) Primitive datatypes like integers and double will default to 0 if uninitialised.



String

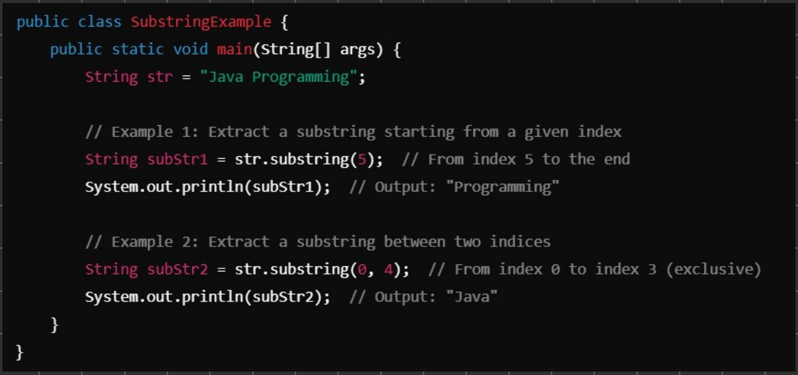
Definition: An object of the string class is used to store a sequence of character.

Confederating (+) The (+) operator can be used to concatenate two strings together to make one string.

int stringA.length() Length() method is a public method which returns the length of a string or an array.

stringA.substring(start index)/(start Index, end Index)

Accessing a sub-string inside a string variable based on index.



```
public class SubstringExample {
    public static void main(String[] args) {
        String str = "Java Programming";

        // Example 1: Extract a substring starting from a given index
        String subStr1 = str.substring(5);  // From index 5 to the end
        System.out.println(subStr1);  // Output: "Programming"

        // Example 2: Extract a substring between two indices
        String subStr2 = str.substring(0, 4);  // From index 0 to index 3 (exclusive)
        System.out.println(subStr2);  // Output: "Java"
    }
}
```

char stringA.charAt(index) Returns the character at the specified index.

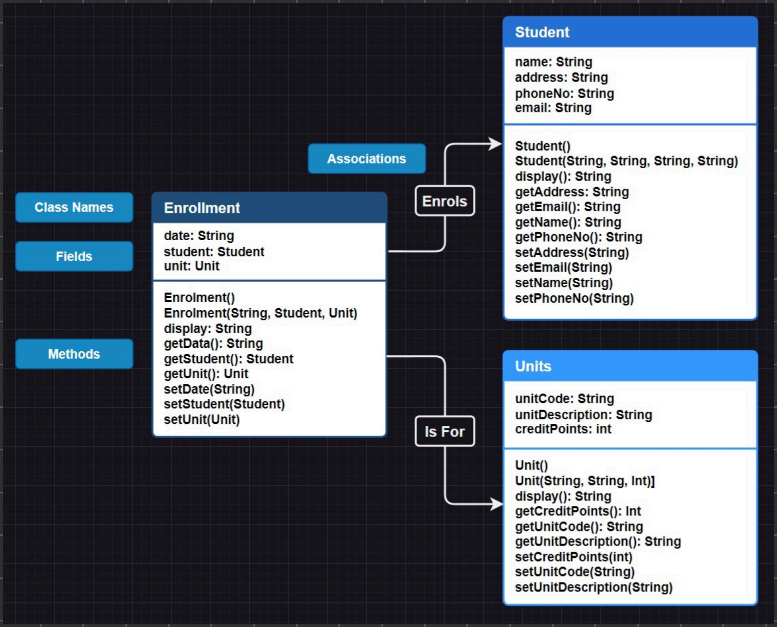
stringA.trim() Returns a copy of the string with the trailing and leading whitespace removed.

stringA.toUpperCase()
stringA.toLowerCase() Returns a copy of the string converted to upper or lower case.

stringA.equals(string1)
stringA.equalsIgnoreCase(string1) Since string is a special object class, the relational operator "==" does not apply. Instead there is a special equality method.

Class Diagrams

In object-oriented programming a common representation is a class diagram from the Unified Modelling Language (UML). The class diagram shows a program's Classes, attributes, methods and relationship. Class diagrams are static representation of how each class interacts.



Responsibility-Driven Design

A design technique where classes are designed based on their behaviours by grouping related methods to be part of the same class with the aim of improving encapsulation.

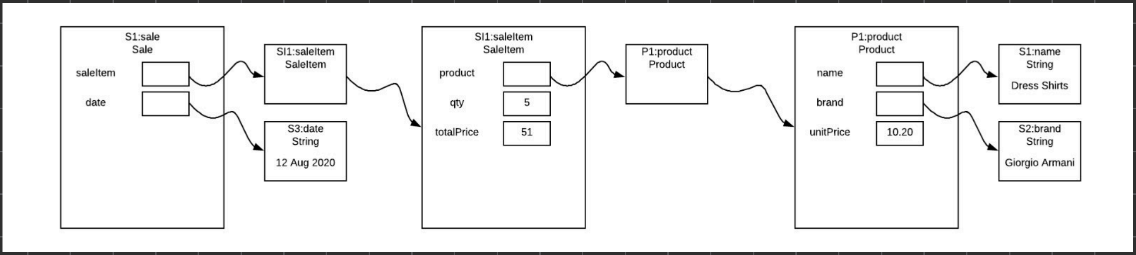
Encapsulation

The idea of wrapping data (fields) and the code that uses the data (methods) together as a single unit.

Object Diagram

An object diagram shows the objects and their relationships at one moment in time (snapshot) during the program execution. It gives information about objects at runtime and presents the dynamic view of a program.

The advantage is that programmers are able to visualise the pass by value and pass by reference aspects.



Documentation: Comments

Comments outline in high level the use of class or method. It is best to outline what it does that rather how the program does it (business logic).

For Class, the should outline what the class does, the author and the version.
For Methods, the comments should outline what it does, the name and of the parameter using tags and a brief description of what it returns.

1) Inline Comments
2) Multi-line Comments
3) Documentation Comments

User Documentation	How to guides, tutorials, reference documentation and explanation documents
Programmer Documentation	API Documentation, release notes, readme documentation and system documentation

Changelog and Test Strategy

Documentation: Javadoc

The documentation for **class**:

1) class name
2) purpose and characteristic of class
3) version number
4) author's
5) documentation for constructor and method

The documentation for **method**:

1) method name
2) return type
3) parameter name and type
4) purpose and function of method
5) description of each parameter
6) description of the value returned

```
javadoc -d <path to output folder> <java source code filename>
```

Tag	Description	Syntax
@author	Defines the author of a class.	@author name-text
{@code}	Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.	{@code text}
{@docRoot}	Represents the relative path to the generated document's root directory from any generated page.	{@docRoot}
@deprecated	Adds a comment indicating that this API should no longer be used.	@deprecated deprecatedtext
@exception	Adds a **Throws** subheading to the generated documentation, with the classname and description text.	@exception class-name description
{@inheritDoc}	Inherits a comment from the **nearest** inheritable class or implementable interface.	Inherits a comment from the immediate surperclass.
{@link}	Inserts an in-line link with the visible text label that points to the documentation for the specified package, class, or member name of a referenced class.	{@link package.class#member label}
{@linkplain}	Identical to {@link}, except the link's label is displayed in plain text than code font.	{@linkplain package.class#member label}
@param	Adds a parameter with the specified parameter-name followed by the specified description to the "Parameters" section.	@param parameter-name description
@return	Adds a "Returns" section with the description text.	@return description
@see	Adds a "See Also" heading with a link or text entry that points to reference.	@see reference
@serial	Used in the doc comment for a default serializable field.	@serial field-description | include | exclude
@serialData	Documents the data written by the writeObject( ) or writeExternal( ) methods.	@serialData data-description
@serialField	Documents an ObjectStreamField component.	@serialField field-name field-type field-description
@since	Adds a "Since" heading with the specified since-text to the generated documentation.	@since release
@throws	The @throws and @exception tags are synonyms. It indicates the exception thrown by the code in that class.	@throws class-name description
{@value}	When {@value} is used in the doc comment of a static field, it displays the value of that constant.	{@value package.class#field}
@version	Adds a "Version" subheading with the specified version-text to the generated docs when the -version option is used.	@version version-text

The **aim** is to **document** your **classes** so that others just need to **read the interface** and **not the implementation**.

Interface describes **what the class can do and how it** can be used in **high level terms**.

Implementation of a class is the **source code** that defines the class.

Documentation: Java API

Java API Link

Java Module

Module: java.base

A collection of Java packages and resources that are group together to form a custom application or API

Java Package

Package: java.lang

Package group related classes together based on the actions they perform.

Java Class

Class: java.lang.String

Class is the set of instructions for the methods that provides the functionality.

StringBuffer

Strings are immutable objects. An immutable object is an object where its contents or state cannot be changed once it has been created.

Say if a method like toUpperCase is used on a s. tring. it does not change the original value of the string, instead it creates a new string. If it is not assigned to a variable, it becomes an anonymous object and could be a vector of attack due to poor memory management.

The StringBuffer (multi-thread safe) & StringBuilder (Unsafe for multi-thread but quicker) classes allow the manipulation of each character of the string. There are methods like append, insert and replace to facilitate the process.