**What is java?**

-Java is a compiled programming language. It is supported by Oracle and allows us to create robust enterprise level applications.

**Why Java?**

**-**Java supports OOP - so we get all the benefits of OOP.

-Compiled - which means that we have really nice debuggins situations.

-WORA - Write Once Run Anywhere, anything that can host the JVM will allow us to run Java code.

-A large list of frameworks and rich documentation - is good!

-C-based : it is easier to pick up

-Garbage Collection - automated memory management.

**What is the JVM? JRE? JDK?**

-JDK: Java Development Kit

Is what we will need to create Java code! The JDK contains

the JRE (and the JVM) and other tools such as the debugger and

the compiler!

-JRE: Java Runtime Environment

Allows us to run Java code. It contains the JVM and core libraries.

-JVM: Java Virtual Machine

Translates Bytecode (machine-readable language) to machine code.

**What is a bit? What is a byte? What is a nibble?**

-Bit – a binary number of 0 or 1 (basic unit of information)

-Byte – 8 bits

-Nibble – 4bits

**What are the naming conventions in java? Classes, packages, etc.**

-Classes: MyClass

-Packages: com.folder.otherfolder

-Methods: myMethod()

-Variables: int i , int myInt

**What are the primitive datatypes in java?**

-int - 32 bits

-char - 16 bits

-boolean - 1 bit (or a byte)

-byte - 8 bits

-short - 16 bits

-double - 64 bits/byte

-float - 32 bits/byte (less precise, but less memory)

-long - 64 bits

**What is flow control?**

-Control flow statements will affect the flow of how your code will be executed.

They allow you to repeat blocks of code or ignore it, depending on what logic

you've implemented.

-**if statement** (can't use break statement)

if([condition that will evaluate to a boolean]){

code block if condition evaluates to true

} else if([condition that will evaluate to a boolean]){

prior if or else if conditions evaluate to false

code block else if condition evaluates to true

} else {

only trigger if no other conditions have been met

}

-**while statement and do while statement**

while([condition that will evaluate to a boolean]){

keeps executing as long as condition evalues to true.

A while loop is very similar to a for loop and is more general.

In fact, you can create a for loop using a while loop.

You can break out of a while loop using "break", which will end the loop at that exact point!

}

**do while** is basically the same as a while loop, but it will execute at least once,

even if the condition has not been met.

**-switch statement**

Why can't it be double?

precision can't be guaranteed. some comparing 2 doubles is very difficult.

instead we can only use int, because they're much easier to compare.

Also true for char and String.

switch([Expression is either a String or a int]){

case [An expression of the same datatype that it will compare the expression to. ]:

This block of code will execute if expression matches.

break; This will escape the entire switch block.

case [An expression of the same datatype that it will compare the expression to. ]:

This block of code will execute if expression matches. Also execute if previous

cases have been met.

default:

Will execute if none of the cases have been met or if one of the cases have been met.

}

**-for statement**

for([starting state]:[Ending state]:[incrementing state]){

Code logic goes;

}

**What does the keyword "break" do? What does the keyword "continue" do? Where can I use it?**

-break; exit out of the looping block , and any code after it (inside of the loop) will not complete!

-continue; breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

-Used in control flow statements [while, do, for, switch]used to break out of loop

**What is short circuiting?**

**-**[**AND(&&)**](https://www.geeksforgeeks.org/operator-in-java-with-examples/)**short circuit:** In case of AND, the expression is evaluated until we get one false result because the result will always be false, independent of the further conditions.

-[**OR(||)**](https://www.geeksforgeeks.org/operator-in-java/)**short circuit:** In case of OR, the expression is evaluated until we get one true result because the result will always be true, independent of the further conditions.

**What datatypes does a switch case allow?**

-precision can't be guaranteed. some comparing 2 doubles is very difficult.

-int, char, String

**What is an array?**

-an array is a list of similar items that can be iterated over.

- Same data types

- Positions are indexed

- Length is immutable

- (Iterable)

-Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

**What method do you call to find the size of an array? Does a method even exist?**

-array.length

**What is a method?**

-A method is a function that is associated with an object.

-methods also follow camelCase

-A method is a block of code which only runs when it is called.

myMethod();

public class Main {

static void myMethod() {

System.out.println("I just got executed!");

}

public static void main(String[] args) {

myMethod();

}

}

**What is a function vs a method?**

**-** . A **method** is a piece of code that is called by a name that is associated with an object.

- A f**unction** is a sub program that does somthing. A block of code that is executed when called upon. It consists of a method signature and body. (allows reuse of blocks of code!)

**What is method overloading?**

- We can use methods with the same name!

-but we have to change the method signature, so that we can specify which method we want!

-We've overloaded by changing:

-the number of arguments/parameters

-the parameter type

-the order

**What is var args in java?**

**-**A method that takes a variable number of arguments is a varargs method.

-A variable-length argument is specified by three periods(…)

example:

public static void fun(int... a)

**What is a class?**

-a class is a 'logical' entity.

- Class is essentially a blueprint for the instantiation of an object. Typically if you want to Access the methods and attributes 'within' a class, you'll need to instantiate it to an object (unless you use a static keyword)

**What is an object?**

-Object is a 'physical' entity,

-Object is an implementation of a class

- Memory is only allocated to the object.

**What are the four pillars of object oriented programming?**

**Inheritance:**

Inheriting the methods and variables from a parent entity

Why?

Promotes code reusue

Java?

We achieve inheritance using the "extends" keyword

or the "implements" keyword, to get class based inheritance.

A child class can inherit methods and attributes from its

parent class/interface.

**Abstraction:**

Concept of hiding implementation to show functionality.

(We hide away impelemntation - how it works- so that we can focus on our

own business logic)

Why?

Allows us to focus on our own level tasks.

Java?

Interfaces (full abstraction) and

Abstract Classes (partial abstraction)

**Polymorphism**

"To take many forms". From inheritance we inherit our parents mehtods

and variables. We're allowed to modify these methods to complete

different actions.

Why?

We get flexibility to have different actions from our behaviours.

Java?

Overriding (runtime polymorhpism) and

Overloading (compile time polymorphsim) !

(What can't you O?)

(What do you need to do to achieve O?)

(Difference betwee Overrding and overloading?)

**Encapsulation**

The pracitice of restricting access to resoucres within your class.

Ensure data isn't accessible without the appropriate permissions.

Why?

I don't want accidental or malicious manipulation of data

within my class!

Java?

public, projected, default (implicit) and private!

**What is a constructor?**

-A constructor is a special method that is used to instantiate an object.

-A constructor DOES NOT HAVE A RETURN TYPE.

-A constructor is called when an object is created, it's invoked by the "new" keyword!

**What does the keyword "new" do in java?**

-The 'new' keyword is a way of instantiating an object into memory

**Can you name 3 types of constructors?**

**-(implicit) DEFAULT constructor**

- Even if a class doesn't have an explicit constructor written, theyre'll be a constructor provided by the JVM during compile time.

-It's pretty much identical to a no-args method signature and it has no body!

-If a constructor already exists, then the JVM won't provide another consturcotr!

**- No-args constructor!**

Car(){

this.colour = "blue";

this.makeAndModel = "Toyota Camry";

this.doors = 4;

this.miles = 100000;

this.years = 10;

//blocks of logic that can be executed!

}

**-** **all-args constructor! (parameterized constructor - allows us to pass in arguments)**

Car(String makeAndModel,int age, int doors, double miles, String colour ){

//object.property = argument

this.makeAndModel = makeAndModel;

this.years = age;

this.doors = doors;

this.miles = miles;

this.colour = colour;

}

**Is the default constructor and the no args constructor the same thing?**

-It's pretty much identical to a no-args method signature and it has no body!

**What is the first implicit line of any constructor?**

-super(); is implicilty called inside of the default constructor!

-super references a parent!

**What is the difference between "super" and "this"?**

-"this" keyword is referencing the object's own attributes!

**What is the difference between initializing and instantiating?**

**-Instantiation**: The new keyword is a Java operator that creates the object.

**-Initialization**: The new operator is followed by a call to a constructor, which initializes the new object.

**What are the scopes of a variable in java?**

-Scope:

Is where a variable exists

Child scopes can reference parent scope variables

-Class scopes:

This is a variable that can be referenced anywhere within the class!

-Object scopes:

Referenced by anthing inside the object.

- Method scopes:

Variables can only be referenced within the method!

-Block/Loop scope

Variables can be only referenced within the block

**What are the access modifiers in java?**

Access modifiers

**public** : this is accessible anywhere, as long as you can reference it!

**protected** : this resource is only available to classes within the same package OR I f it is a child class!

(**default**) : this resource is only available to classes within the same package

**private** : the variable or the method, is only accessible within the class

**What is shadowing?**

Shadowing in Java is the practice of using variables in overlapping scopes with the same name where the variable in low-level scope overrides the variable of high-level scope.

**What is the difference between overloading and overriding?**

**Overriding:**

@Override

-Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, same parameters or signature, and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to *override* the method in the super-class.

-When overriding, the return type has to match

-Method signature has to have the same number of arguments!

-If you change arguments, you're effectively overloading!

**What is runtime polymorphism? What is compile time polymorphism?**

**-Compile Time Polymorphism:** Whenever an object is bound with their functionality at the compile-time, this is known as the compile-time polymorphism. At compile-time, java knows which method to call by checking the method signatures. So this is called compile-time polymorphism or static or early binding. Compile-time polymorphism is achieved through [method overloading](https://www.geeksforgeeks.org/overloading-in-java/).

**-Run-Time Polymorphism:** Whenever an object is bound with the functionality at run time, this is known as runtime polymorphism. The runtime polymorphism can be achieved by [method overriding](https://www.geeksforgeeks.org/overriding-in-java/).

**What is type casting?**

**-**Type casting is when you assign a value of one primitive data type to another type.

**What is the difference between up and down casting?**

2 types of casting

Widening Casting-Widening casting is done automatically when passing a smaller size type to a larger size type:

byte -> short -> char -> int -> long -> float ->double

Narrowing Casting-Narrowing casting must be done manually by placing the type in parentheses in front of the value:

double -> float -> long -> int -> char -> short - > byte

**What is a String?**

-String

String is an object, that contains an array of characters!

It allows us to create and manipulate strings.

-The string class is immutable and the class is final.

immutable: the string can't be changed!

final: the class cannot be extended.

Strings cannot be modified, instead "modified" versions are created in memory

This is to stop other reference variables from being affected.

- String "Objects" are stored in the heap but String literals are stored in the String pool.

(Which is also within the heap)

**What is the difference betwen String, StringBuilder, and StringBuffer?**

**StringBuilder:**

Since strings are immutable, they're inefficient to use if we keep changing the string

StringBuilder is a mutable version of String

**StringBuffer:**

StringBuffer is basically a StringBuilder, but with lower performance!

It is 'synchronized' - it is thread safe. So you can do multi-threading operations with it.

**Difference between Collection vs Collections?**

**-Collection** is a [interface](https://www.geeksforgeeks.org/interfaces-in-java/) present in java.util.package. It is used to represent a group of individual objects as a single unit.

**-Collections** is a utility class present in java.util.package. It defines several utility methods like sorting and searching which is used to operate on collection. It has all static methods.

**What are the 3 child interfaces in the Collection hiearchy?**

**What are the child classes of the collection hiearchy?**

Diagram

Description automatically generated

**\* List:**

\* Each position on a list is indexed

\* List keep track of insertion order

\* Duplicate elements

\*

\* (vs Array : we get flexible length as well more methods

\*

\*/

**//GENERICS <>**

//"<>" are generics, taht enforce type safety.

//I can have a list of any objects, but I can narrow it down to my objects, by adding in generics to limit

**//LIST**

//flexible in size, it will automatically resize depending on how many elements there are

/\*

\*

\* An ordered collection

\* May contain duplicate leements

\* Access via index

\*

\* Lists have to contain the same elements in teh same position to be equal!

\*

**\* ArrayList:**

\* A resizable array (mutable) compared to the buld in array inside Java. Allows you to add and remove elements.

\* (Almost always the better performance)

\*

**\* LinkedLists:**

\* Consists of nodes, each node will store a value a 'pointer' to the next node.

\* (Adding or removing a lot of stuff at a time, then linked list has a little bit more performance!)

\*

**\* Vector**

\* Depcreated (contains legacy methods), but is threadsafe.

\*/

**//Queue**

// Ordered list of objects, by default is related to insertion order.

// Follows the FIFO (First in First out) and also LILO (Last in last out) principles

// methods: pool, remove, peek and element!

/\*

\*

**\* LinkedList:**

\* Same as from list, but cast into queue

\*

**\* PriorityQueue:**

\* Doesn't permit null

\* Doesn't permit hetreogenous object (object's that it can't compare)

\*/

**//Set**

/\*

\* Does not allow duplicates

\* Does not guarantee insertion order

\* Set is equal if they contains the same elements.

\*/

**Collection API**

-Called a framework (more akin to a library). Provides interfaces and classes that allow developers to more easily manage groups of objects!. A 'collection' object is designed to store a group of objects. (Gives you a lot more flexibility than the array object)

**Why?**

-Array object within Java is limited to work with. Mainly, the array is immutable. I could create my own custom version of a flexible array BUT it'll likely by poorly optimised!

-Secondly, it's difficlut to share data strcutures between developers!

-Standardised and optomised data structures!

**Advantages**

-Reduced effort (provides data structures and algorithms for you)

-Increases performance (more optomised)

-Encourages software reuse (provides a standard interface)

**Comparator vs Comparable interface?**

**What primitives can I store in my Collection?**

**-Cant**

**Is Map in the Collection framework? How is it different from the rest?**

-not a subtype of the [Collection interface](https://www.geeksforgeeks.org/collections-in-java-2/).

-Maps work with key/value pairs, while the other collections work with just values. Map maps keys to values. It allows its content to be viewed as a set of keys, a collection of values and a set of key-value mappings.

**What are the child classes of Map?**

**-HashMap** is a part of Java’s collection since Java 1.2. It provides the basic implementation of the Map interface of Java. It stores the data in (Key, Value) pairs. To access a value one must know its key. This class uses a technique called [Hashing](http://www.geeksforgeeks.org/hashing-data-structure/). Hashing is a technique of converting a large String to small String that represents the same String. A shorter value helps in indexing and faster searches. Let’s see how to create a map object using this class.

 -**LinkedHashMap** is just like HashMap with an additional feature of maintaining an order of elements inserted into it. HashMap provided the advantage of quick insertion, search and deletion but it never maintained the track and order of insertion which the LinkedHashMap provides where the elements can be accessed in their insertion order.

-**The TreeMap** in Java is used to implement Map interface and NavigableMap along with the Abstract Class. The map is sorted according to the natural ordering of its keys, or by a Comparator provided at map creation time, depending on which constructor is used. This proves to be an efficient way of sorting and storing the key-value pairs. The storing order maintained by the treemap must be consistent with equals just like any other sorted map, irrespective of the explicit comparators.

**What is a wrapper class?**

**What is autoboxing? What is unboxing?**

//Wrapper Classes

//Allow us to 'wrap' our primitives, so that tey become objects!

///Boxing is the process of converting a primitive to its wrapper class!

// Autoboxing! This is implicilty done by the class, automatically!

**What is Inheritance? How would you describe it to a C++ developer?**

**How does Java Implement inheritance?**

Inheritance:

One of the pillars of OOP

Is the process of inheritng properties and methods

We use the "extends" keyword to inherit

e.g. Truck extends Car, Truck is the child class of CAr.

Truck may have access to all of the Car's properties and methods

The Car, is a parent, so won't have access to the Truck's methods or porperties!

For homogeneous inheritance we use the keyword: "extends"

class -> class

abstract class -> abstract class

abstact class -> class

interface -> interface

For hetrogeneous inheritance we use the keyword: "implements"

interface -> class

interface -> abstract class

Child class have access to their parent class, e.g. (multilevel inheritance)

A -> B -> C

C will have access to all the methods and variables in B and A.

B will have access to all the methods and variables in A (But not C)

A will not have access to B's and C's methods and variables;

In Java classes cannot extend multiple classes.

**Interface vs Abstract Class vs Class**

**\* Interface:**

Cannot have any concerte methods\*\*\* (all methods are abstract)

Default all methods are 'abstract' and public.

variables are by default defined as public, static and final.

An inteface cannot contain a constructor!

You can implement multiple interfaces!

**\* What is an abstract class?**

Partial abstraction

You can have concrete methods in an abstract class! (unlike an interface\*\*\*)

An abstract class doesn't need an abstract method!

(Abstract class and a class act very similar)

A class cannot extend multiple classes!

Abstract classes can also have constructors!

Abstract Class is a class, that can't be instantiated and may contain abstract methods!

**What is polymorphism?**

**How does Java Implement polymorphism?**

**Polymorphism**

"To take many forms". From inheritance we inherit our parents mehtods

and variables. We're allowed to modify these methods to complete

different actions.

Why?

We get flexibility to have different actions from our behaviours.

Java?

Overriding (runtime polymorhpism) and

Overloading (compile time polymorphsim) !

**What is Encapsulation?**

**How does Java achieve encapsulation?**

**Encapsulation**

The pracitice of restricting access to resoucres within your class.

Ensure data isn't accessible without the appropriate permissions.

Why?

I don't want accidental or malicious manipulation of data

within my class!

Java?

public, projected, default (implicit) and private!

**What is Abstraction?**

**Abstraction:**

Concept of hiding implementation to show functionality.

(We hide away impelemntation - how it works- so that we can focus on our

own business logic)

Why?

Allows us to focus on our own level tasks.

Java?

Interfaces (full abstraction) and

Abstract Classes (partial abstraction)