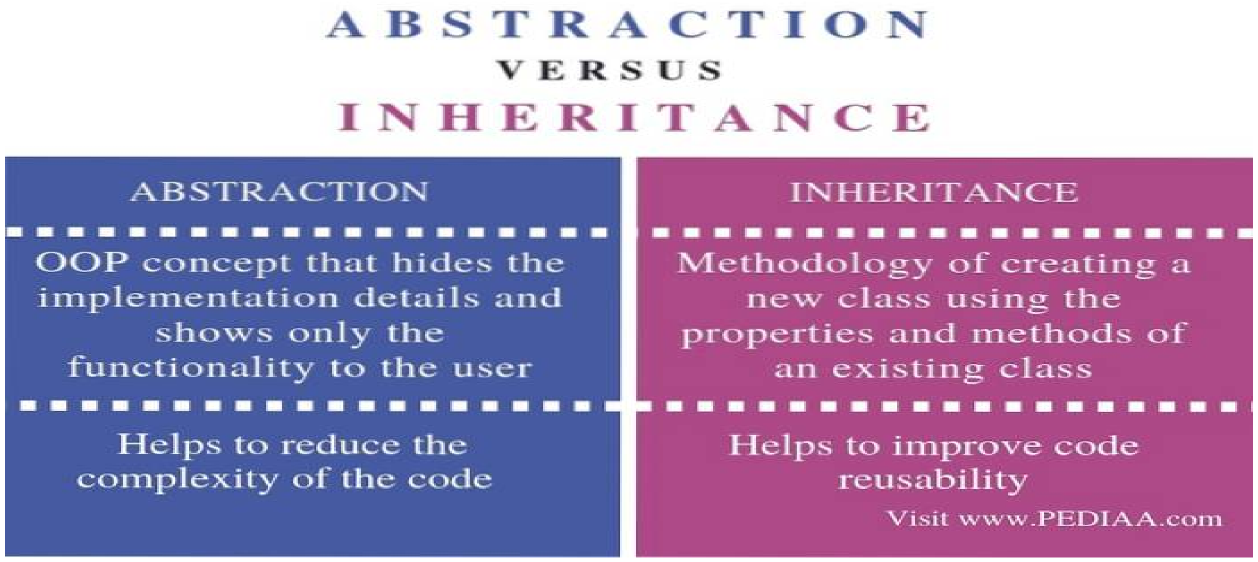
Abstraction	Encapsulation
Abstraction is the method of **hiding unwanted information.**	Encapsulation is a method to **hide the data(variables) and code(methods) together into a single entity or unit** to protect it from outside.
In abstraction, problems are solved at the **design or interface level.**	While in encapsulation, problems are solved at the **implementation level.**
It focuses on the **external** lookout.	It focuses on **internal** working.
It is the process of **gaining the information**	It is the process of **containing the information**
It can be implemented using **abstract classes and interfaces.**	It can be implemented by using the **access modifiers (private, public, protected).**
In abstraction, **implementation complexities are hidden using abstract classes and interfaces.**	While in encapsulation, the **data is hidden using methods of getters and setters.**
The objects are **encapsulated** that helps to perform abstraction.	The object need not to **abstract** that result in encapsulation.



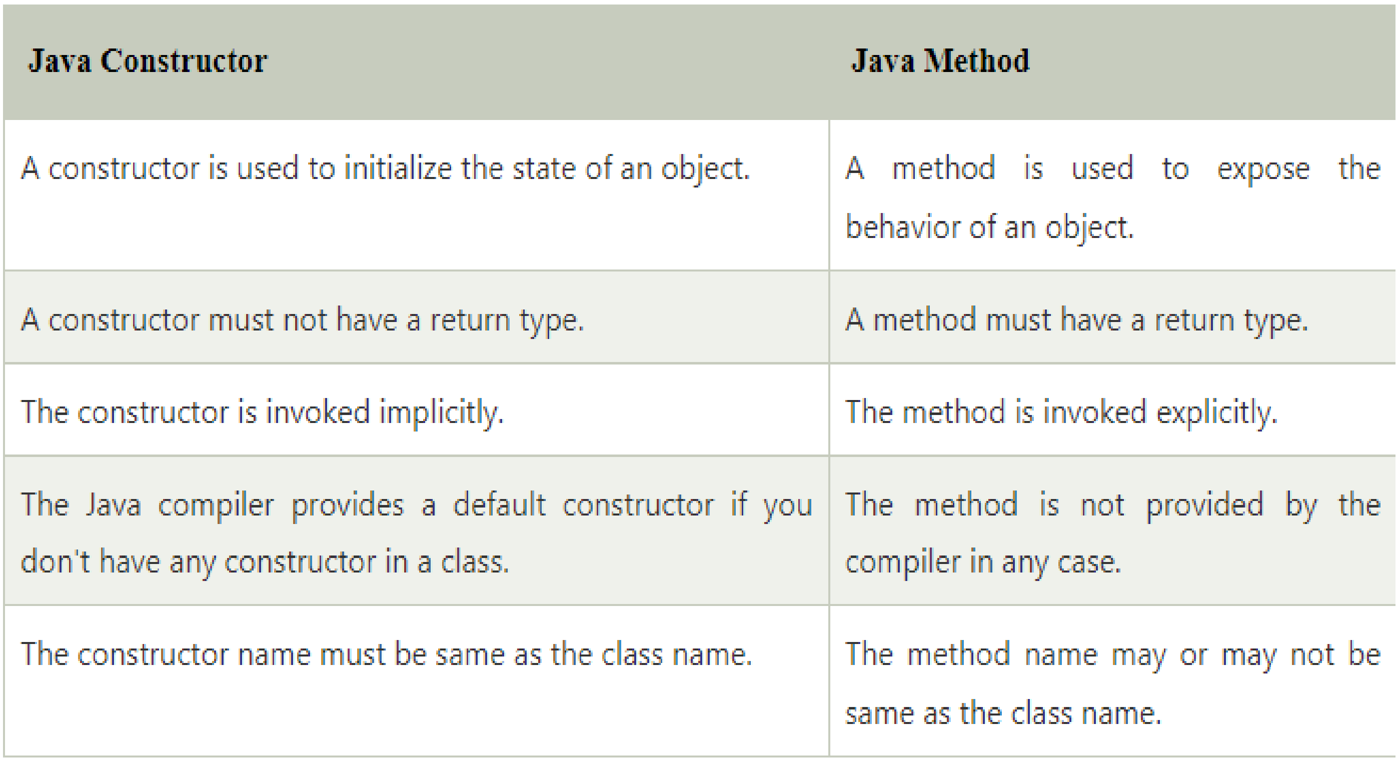
ABSTRACTION VERSUS INHERITANCE

ABSTRACTION	INHERITANCE
OOP concept that hides the implementation details and shows only the functionality to the user	Methodology of creating a new class using the properties and methods of an existing class
Helps to reduce the complexity of the code	Helps to improve code reusability

Visit www.PEDIAA.com

ABSTRACT CLASS IN JAVA
VERSUS
INTERFACE IN JAVA

ABSTRACT CLASS IN JAVA	INTERFACE IN JAVA
A class declared with an abstract keyword, which is a collection of abstract and non-abstract methods	An interface in Java is a reference type that is similar to a class that is a collection of abstract methods
Can have final, non-final, static and non-static variables	Can only have static and final variables
Can have abstract methods and non-abstract methods	Can only have abstract methods
Cannot be used to implement multiple inheritance	Can be used to implement multiple inheritance
Declared using the abstract keyword	Declared using the interface keyword
Can be extended using the keyword "extends"	Can be implemented using keyword "implements"
Can be implemented using keyword "implements"	Used to implement abstraction as well as multiple inheritance



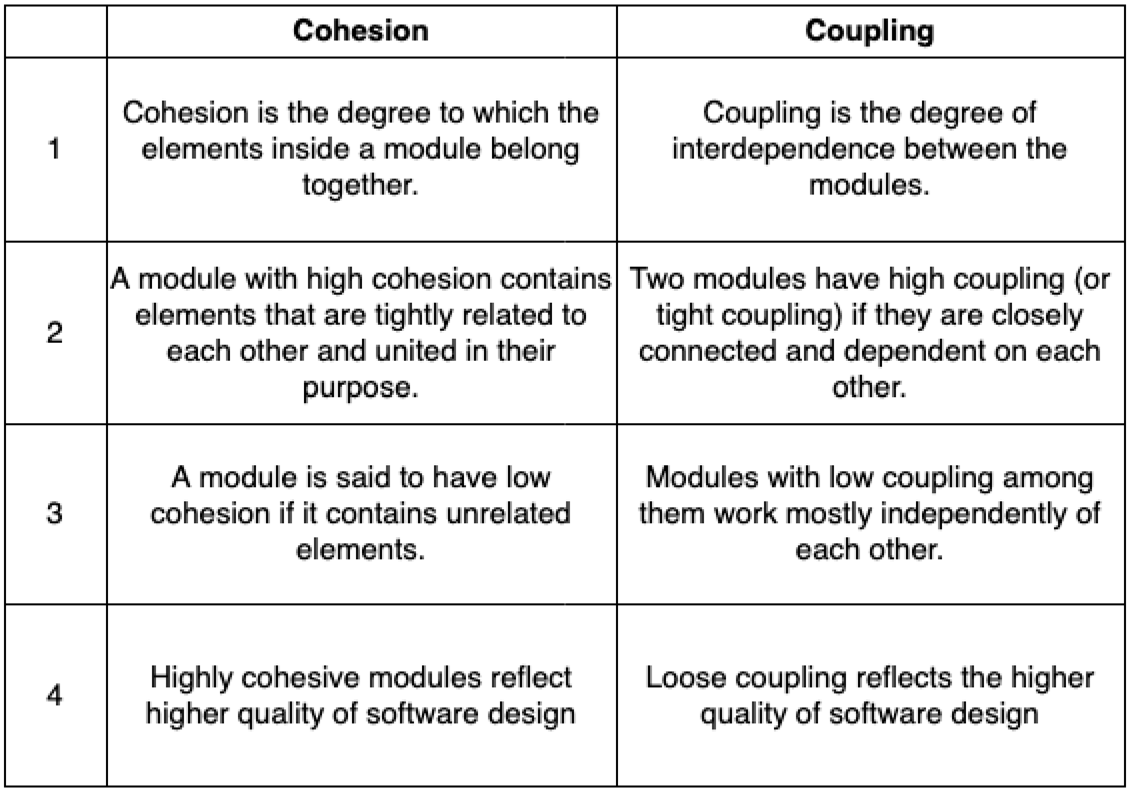
Java Constructor	Java Method
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor name must be same as the class name.	The method name may or may not be same as the class name.

this keyword	**super keyword**
The `this` keyword points to a reference of the current class	the `super` keyword points to a reference of the parent class.
`this` can be used to access variables and methods of the current class	`super` can be used to access variables and methods of the parent class from the subclass.
The `this` keyword is commonly used when an instance variable is shadowed by a parameter of a method	The `super` keyword is useful when a class overrides a method of its parent class, and we need to invoke the overridden method

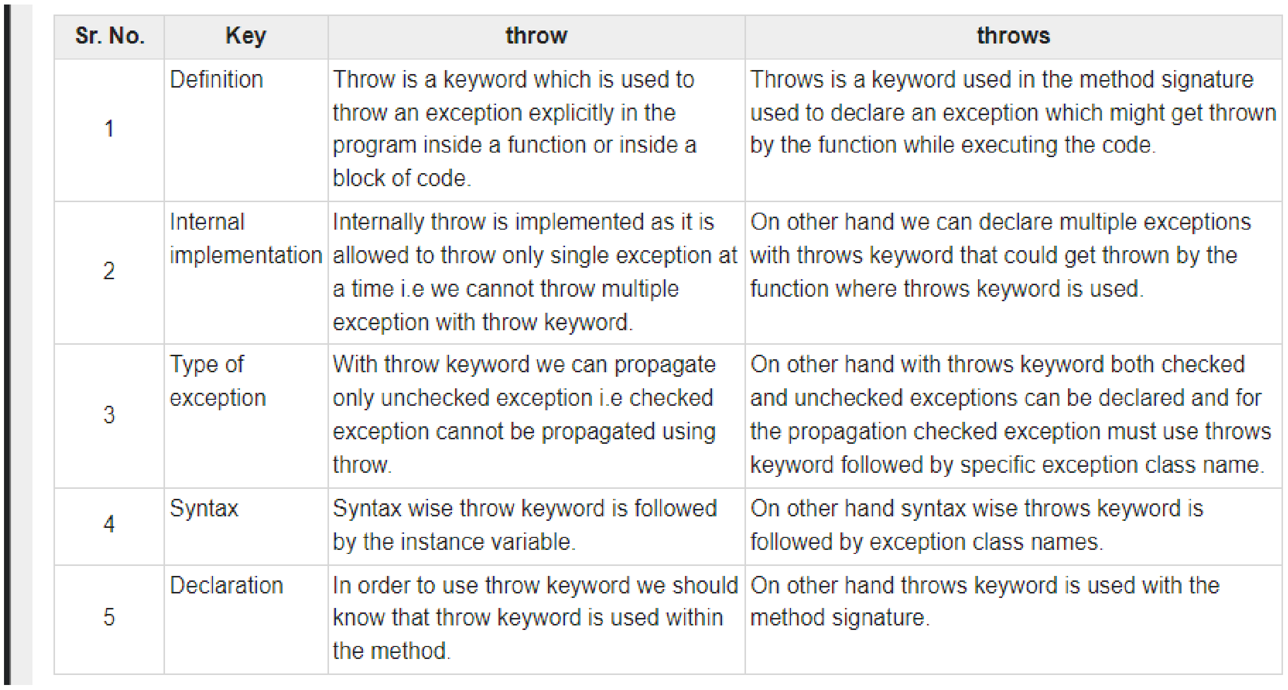
this()	super()
1. this() represents the current instance of a class	1. super() represents the current instance of a parent/base class
2. Used to call the default constructor of the same class	2. Used to call the default constructor of the parent/base class
3. Used to access methods of the current class	3. Used to access methods of the base class
4. Used for pointing the current class instance	4. Used for pointing the superclass instance
5. Must be the first line of a block	5. Must be the first line of a block

S.No	Upcasting	Downcasting
1.	A child object is typecasted to a parent object.	The reference of the parent class object is passed to the child class.
2.	We can perform Upcasting implicitly or explicitly.	Implicitly Downcasting is not possible.
3.	In the child class, we can access the methods and variables of the parent class.	The methods and variables of both the classes(parent and child) can be accessed.
4.	We can access some specified methods of the child class.	All the methods and variables of both classes can be accessed by performing downcasting.
5.	Parent p = new Parent()	Parent p = new Child() Child c = (Child)p;

Static Binding	Dynamic Binding
When the type of the object is determined at **compile time** it is known as Static binding.	When the type of the object is determined at **run-time** it is known as Dynamic binding.
static binding uses type of class to bind	dynamic binding uses type of object to bind
faster	slower
overloaded methods are bonded using static binding	overridden methods are bonded using dynamic binding.

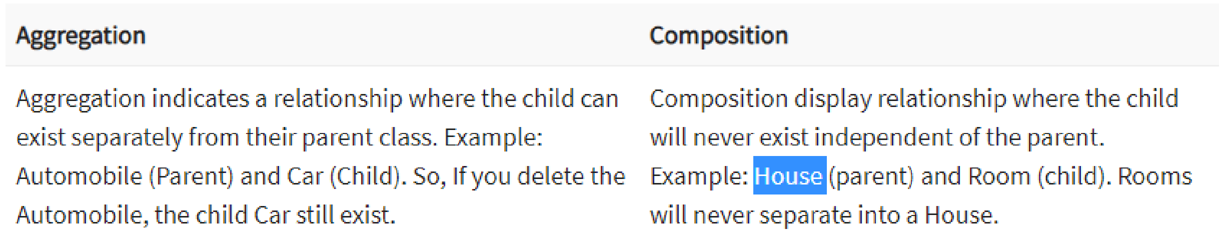


	Cohesion	Coupling
1	Cohesion is the degree to which the elements inside a module belong together.	Coupling is the degree of interdependence between the modules.
2	A module with high cohesion contains elements that are tightly related to each other and united in their purpose.	Two modules have high coupling (or tight coupling) if they are closely connected and dependent on each other.
3	A module is said to have low cohesion if it contains unrelated elements.	Modules with low coupling among them work mostly independently of each other.
4	Highly cohesive modules reflect higher quality of software design	Loose coupling reflects the higher quality of software design



Sr. No.	Key	throw	throws
1	Definition	Throw is a keyword which is used to throw an exception explicitly in the program inside a function or inside a block of code.	Throws is a keyword used in the method signature used to declare an exception which might get thrown by the function while executing the code.
2	Internal implementation	Internally throw is implemented as it is allowed to throw only single exception at a time i.e we cannot throw multiple exception with throw keyword.	On other hand we can declare multiple exceptions with throws keyword that could get thrown by the function where throws keyword is used.
3	Type of exception	With throw keyword we can propagate only unchecked exception i.e checked exception cannot be propagated using throw.	On other hand with throws keyword both checked and unchecked exceptions can be declared and for the propagation checked exception must use throws keyword followed by specific exception class name.
4	Syntax	Syntax wise throw keyword is followed by the instance variable.	On other hand syntax wise throws keyword is followed by exception class names.
5	Declaration	In order to use throw keyword we should know that throw keyword is used within the method.	On other hand throws keyword is used with the method signature.

Aggregation vs. Composition



Aggregation	Composition
Aggregation indicates a relationship where the child can exist separately from their parent class. Example: Automobile (Parent) and Car (Child). So, If you delete the Automobile, the child Car still exist.	Composition display relationship where the child will never exist independent of the parent. Example: House (parent) and Room (child). Rooms will never separate into a House.

ARRAY
VERSUS
ARRAYLIST

ARRAY	ARRAYLIST
A data structure consisting of a collection of elements each identified by the array index	A class that supports dynamic arrays which can grow as needed
A part of core Java programming	A part of Collection framework with other classes such as Vector, HashMap, etc.
Programmer can use the assignment operator to store elements into the array	Programmer can use the add method to insert elements
Can contain primitives or objects	Can only store objects
Helps to implement a fixed size data structure	Helps to implement dynamic size arrays

Visit www.PEDIAA.com