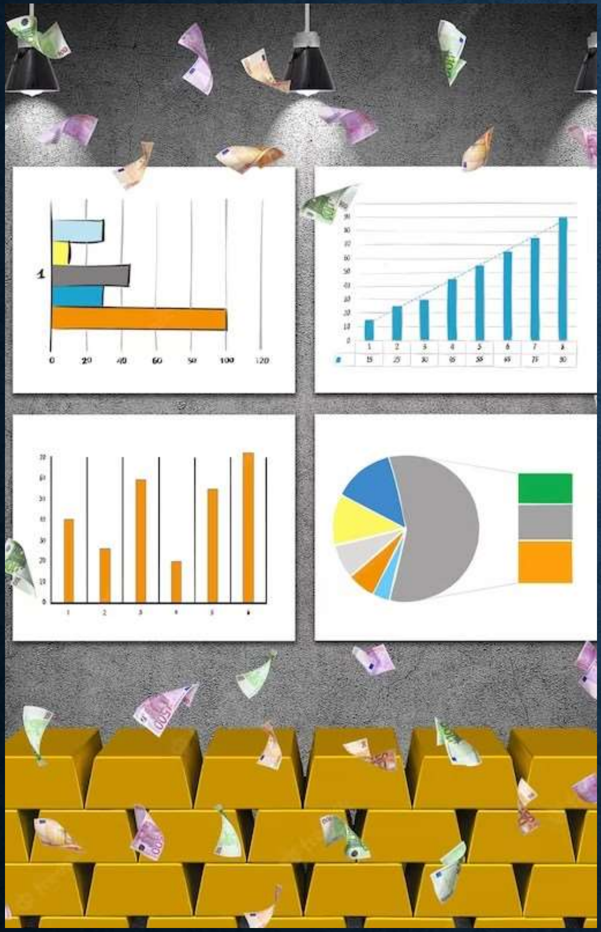
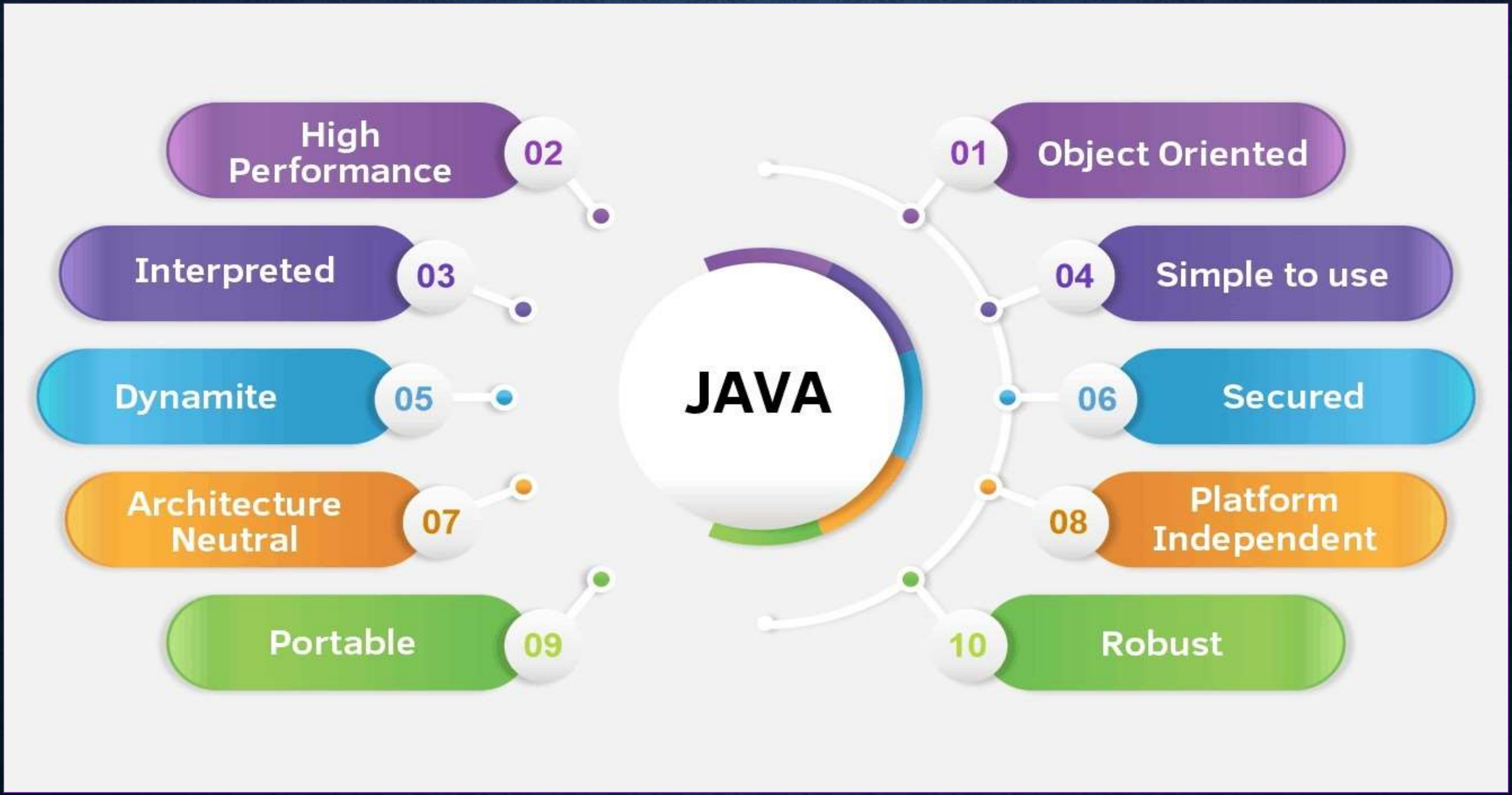


Object-Oriented Programming in Java: Unleashing the Power of OOP

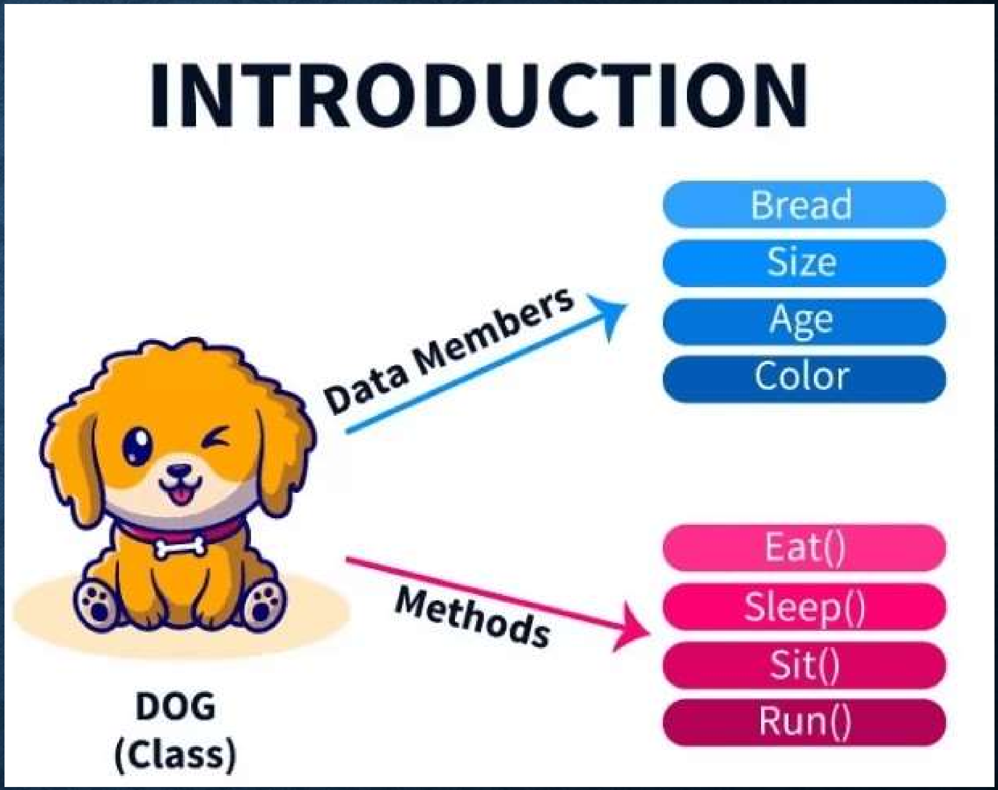
DJS Compute powered by aws

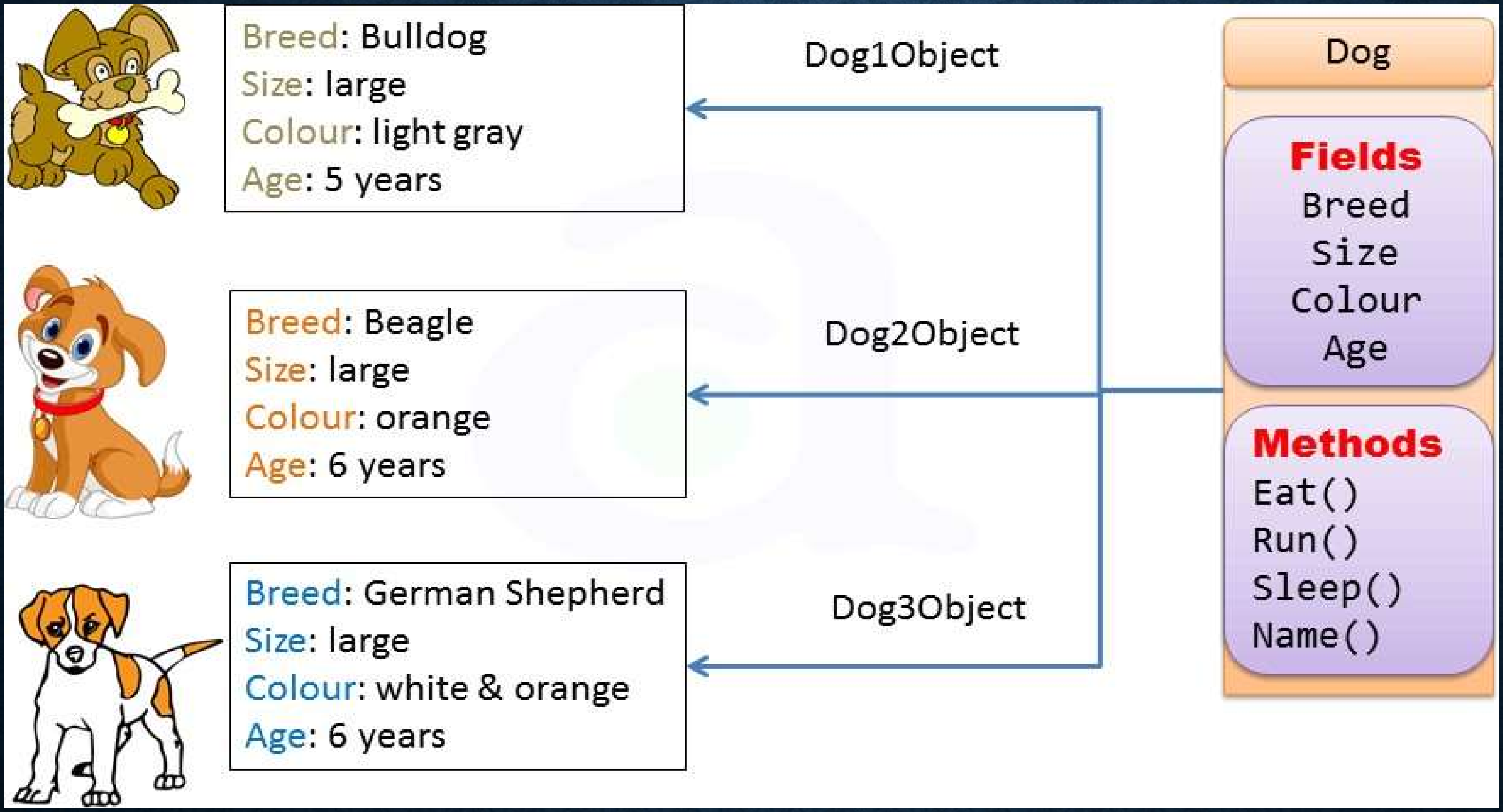

JAVA
01 Object Oriented
02 High Performance
03 Interpreted
04 Simple to use
05 Dynamite
06 Secured
07 Architecture Neutral
08 Platform Independent
09 Portable
10 Robust

UNDERSTANDING OBJECTS AND CLASSES

Objects are the basic run time entities in an object- oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle.

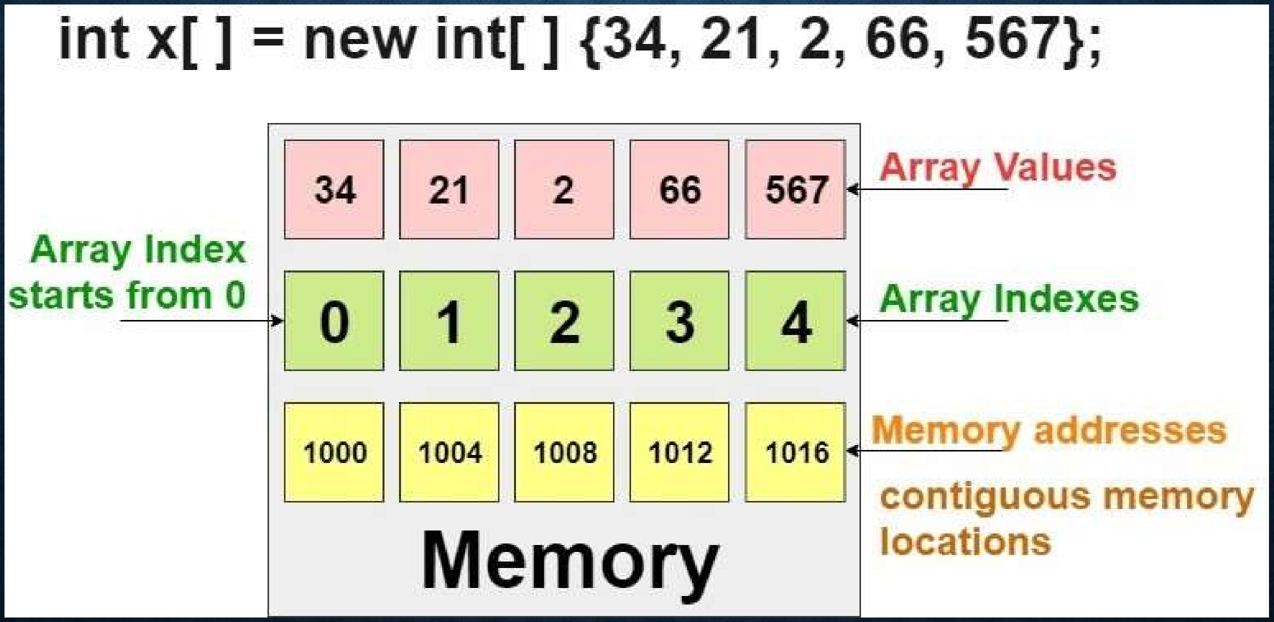
In fact, Objects are variables of the type class. Once a class has been defined, we can create any number of objects belonging to that class.




Breed: Bulldog
Size: large
Colour: light gray
Age: 5 years
Dog1Object
Breed: Beagle
Size: large
Colour: orange
Age: 6 years
Dog2Object
Breed: German Shepherd
Size: large
Colour: white & orange
Age: 6 years
Dog3Object
Dog
Fields
Breed
Size
Colour
Age
Methods
Eat()
Run()
Sleep()
Name()

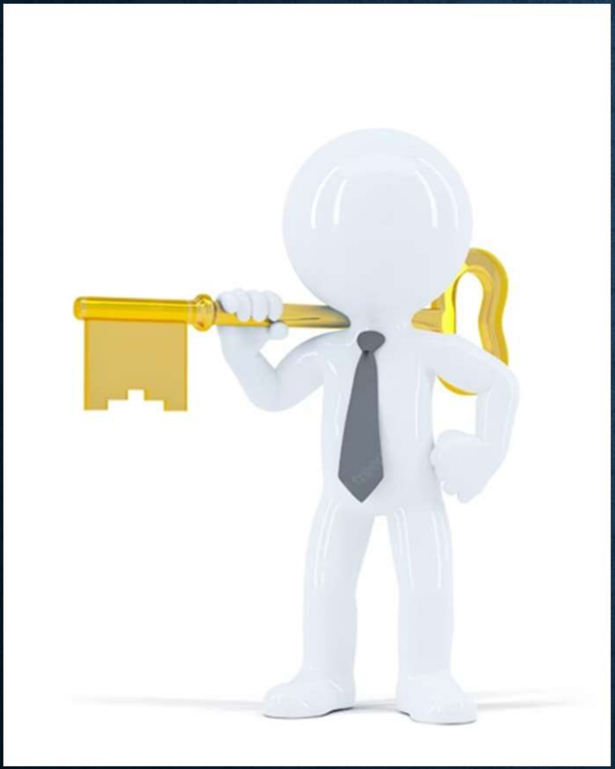
Arrays in java

An array is a collection of similar type of elements which has contiguous memory location.


int x[ ] = new int[ ] {34, 21, 2, 66, 567};
34
21
2
66
567
Array Values
Array Index starts from 0
0
1
2
3
4
Array Indexes
1000
1004
1008
1012
1016
Memory addresses
contiguous memory locations
Memory

ENCAPSULATION:

PROTECTING DATA AND BEHAVIOR



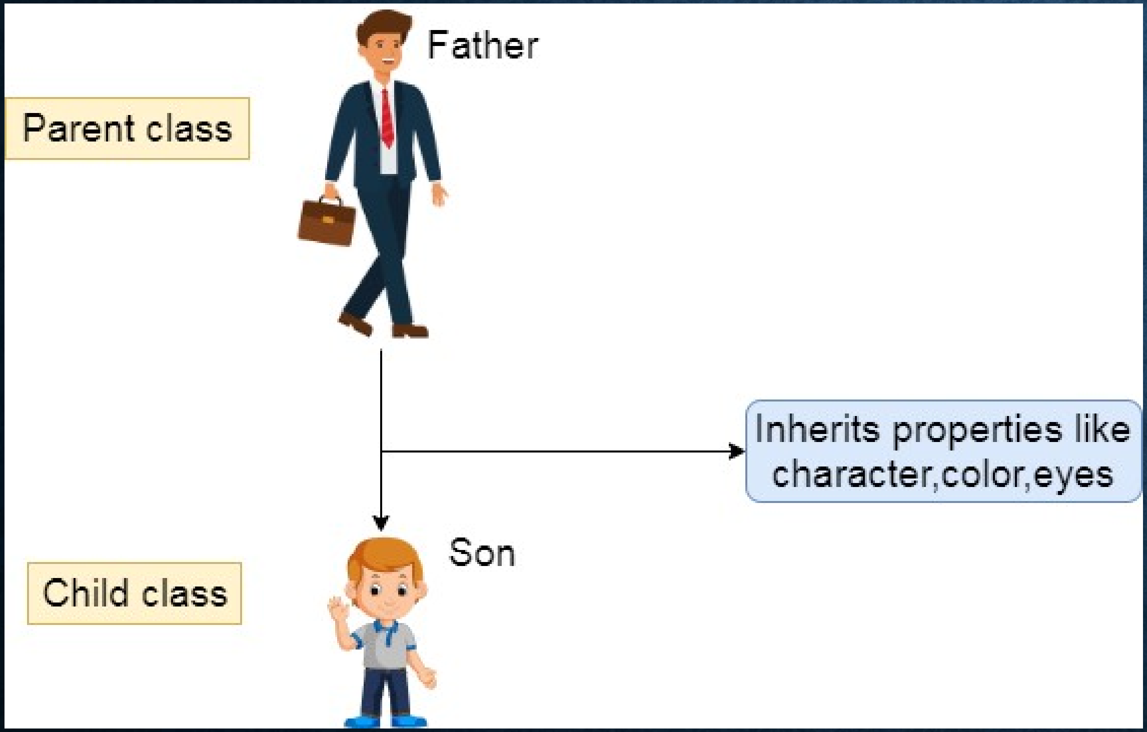
Encapsulation allows you to hide data and methods within a class, ensuring **data integrity** and **code reusability**.
Discover the power of encapsulation and learn how to design classes that protect and efficiently manage data.

ACCESS MODIfiERS: PUBLIC, PRIVATE, PROTECTED

1. **Public:** keyword applied to a class, makes it available/visible everywhere. Applied to a method or variable, completely visible.
2. **Private :** fields or methods for a class only visible within that class. Private members are not visible within subclasses, and are not inherited.
3. **Protected :** members of a class are visible within the class, subclasses and also within all classes that are in the same package as that class

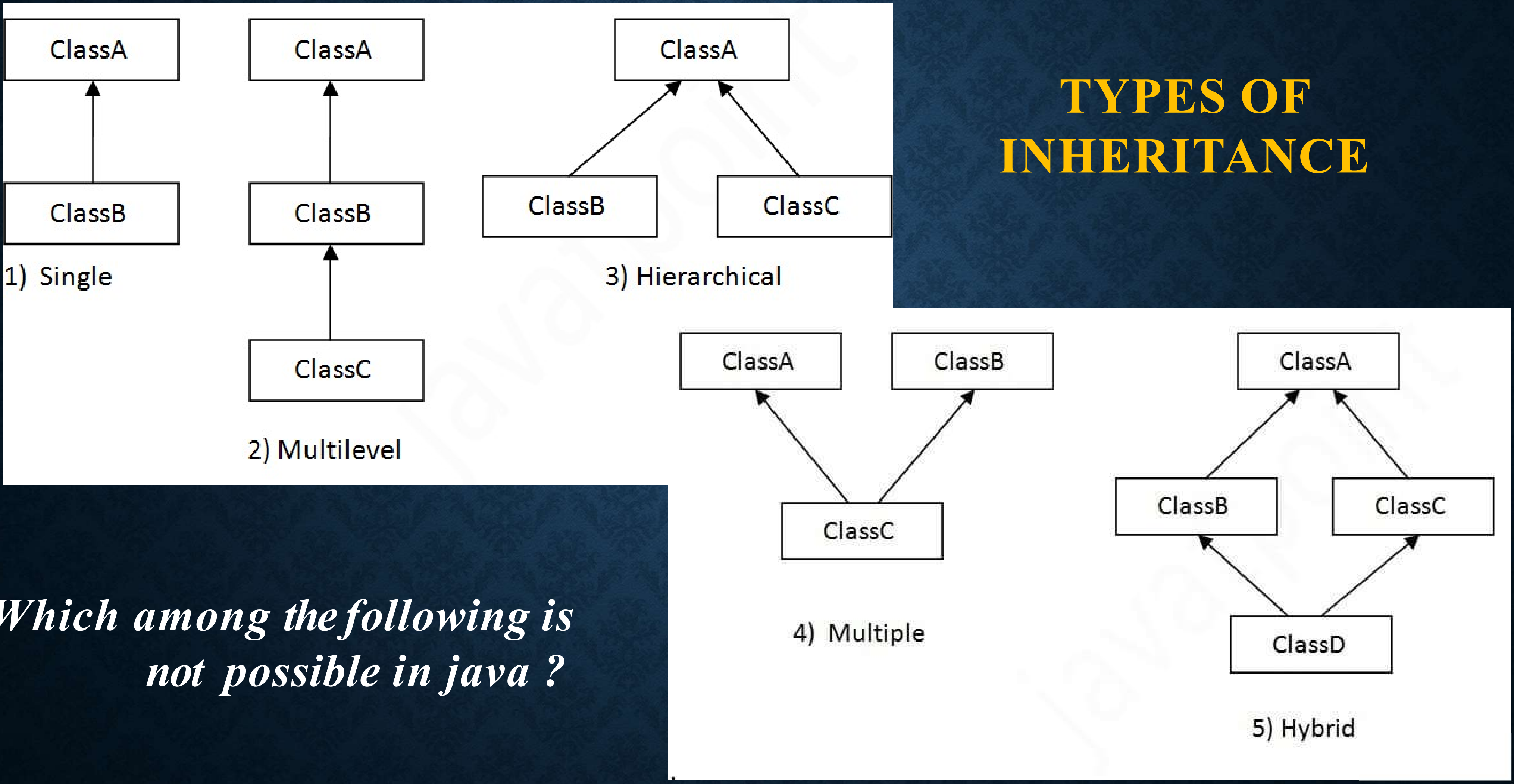
```
public class Circle
{
private double x,y,r;
{
this.x = x; this.y = y; this.r = r;
} //Methods to return circumference and area
public double circumference() {
return 2*3.14*r;
}
public double area() {
return 3.14 * r * r;
}
}
```

INHERITANCE: BUILDING HIERARCHIES OF CLASSES



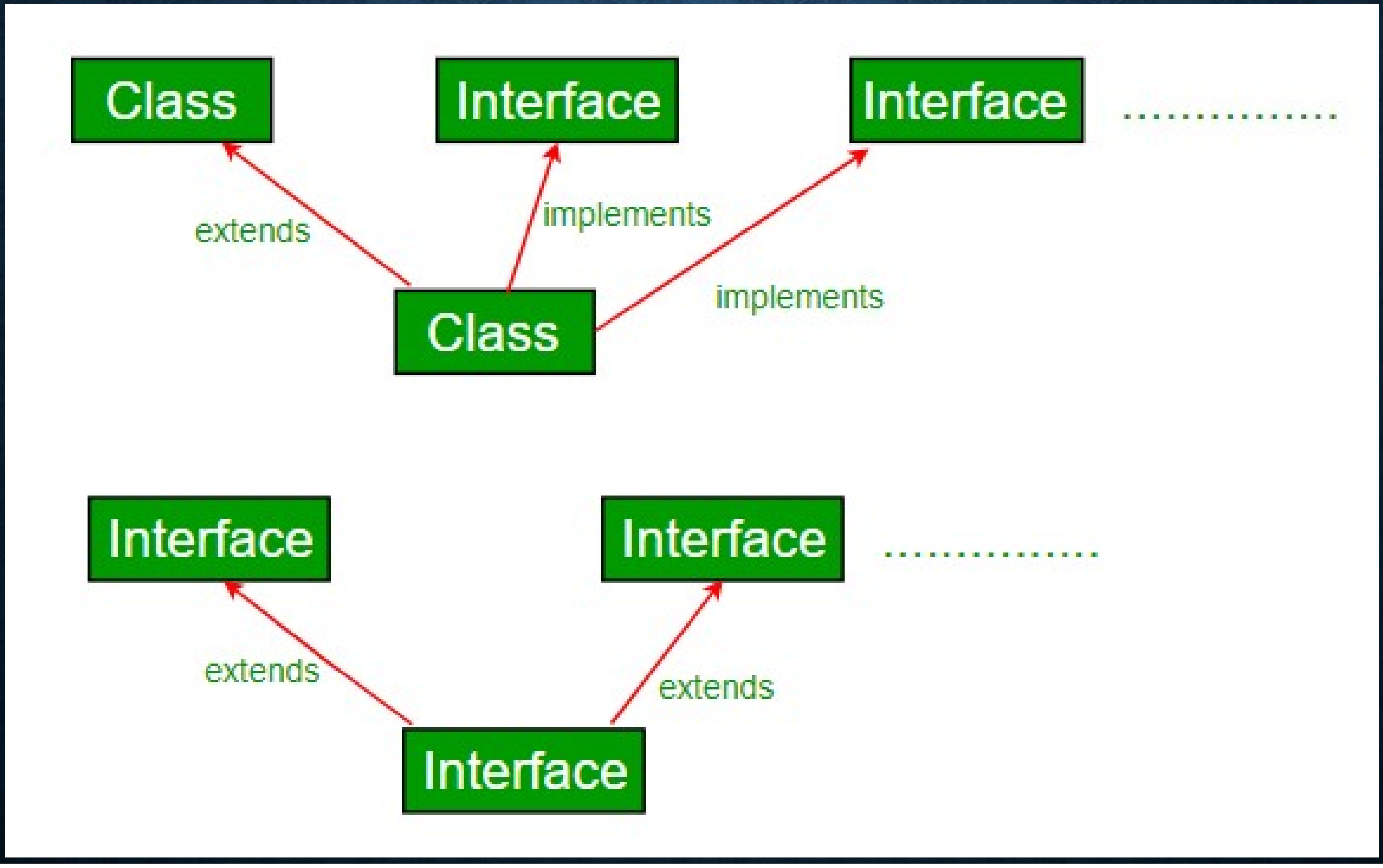
Inheritance enables the creation of **class hierarchies** and **reusability** of code. Explore how to establish **parent-child relationships** between classes, leverage **inheritance benefits**, and effectively design class hierarchies.

TYPES OF INHERITANCE



Which among the following is not possible in java ?

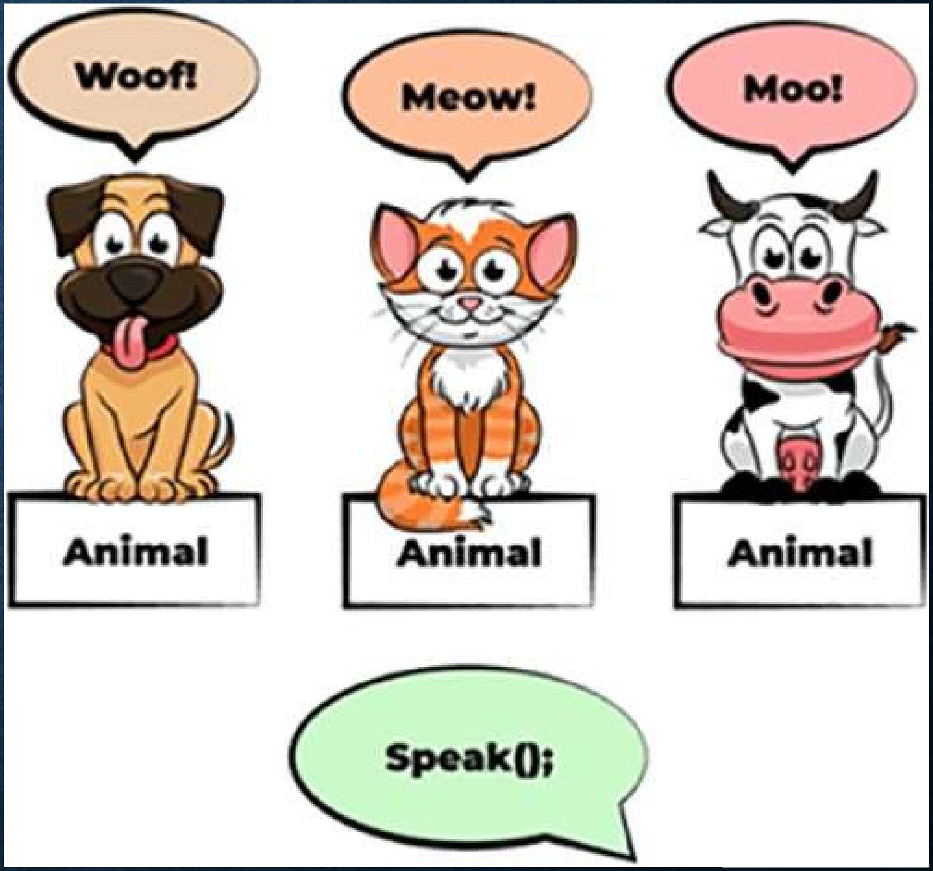
MULTIPLE INHERITANCE (THROUGH INTERFACES)



POLMORPHISM

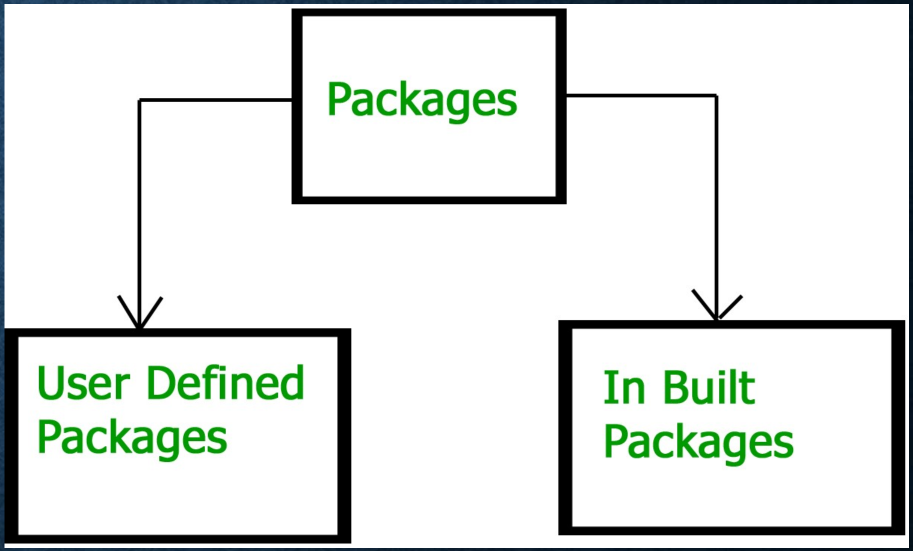
Polymorphism in Java is a concept by which we can perform a single action in diIerent ways. Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

There are two types of polymorphism in Java: compile-time polymorphism and runtime polymorphism.
We can perform polymorphism in java by method overloading and method overriding.



PACKAGES

1. A java package is a group of similar types of classes, interfaces and sub-packages.
2. Package in java can be categorized in two form, built-in package and user-defined package.
3. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.



Define Package MyPack

```
package MyPack;
public class Balance {
String name;
double bal;
public Balance(String n, double b) {
name = n; bal = b;
}
public void show() {
if (bal<0)
System.out.print("-→> ");
System.out.println(name + ": $" +
bal); }
```

IMPORTING MYPACK

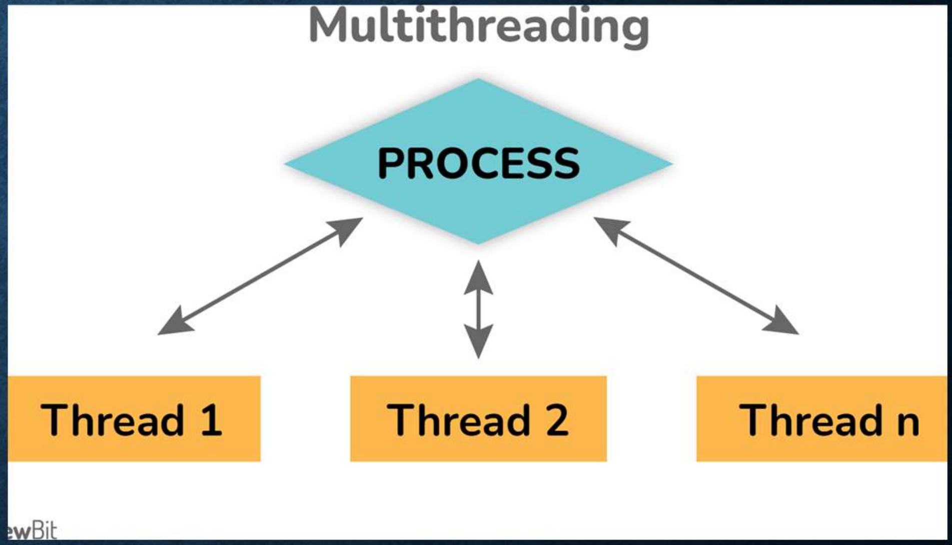
```
import MyPack.*;
class TestBalance {
public static void main(String args[]) {
Balance test = new Balance("J. J. Jaspers",
99.88); test.show();
}
}
```

MULTI-THREADING

Multi-threading enables to write e cient programs that make the ma imum use of the CPU, keeping the idle time to a minimum.

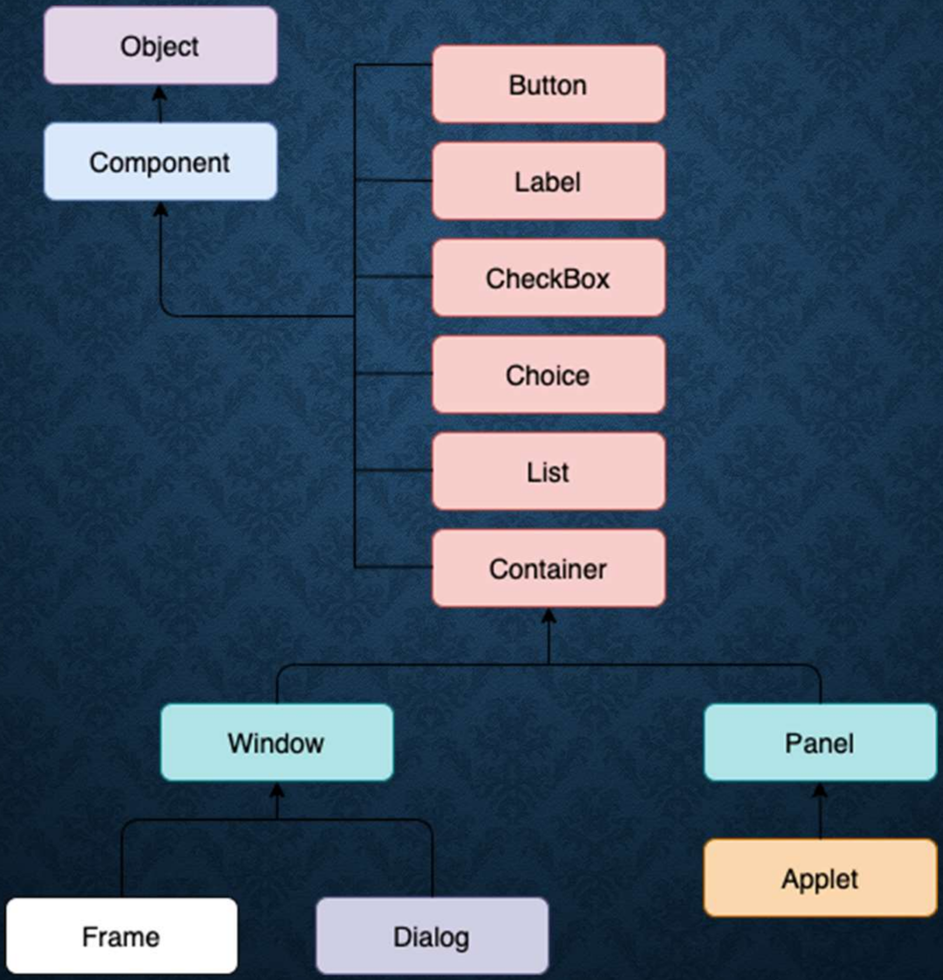
There is plenty of idle time for interactive, networked applications:

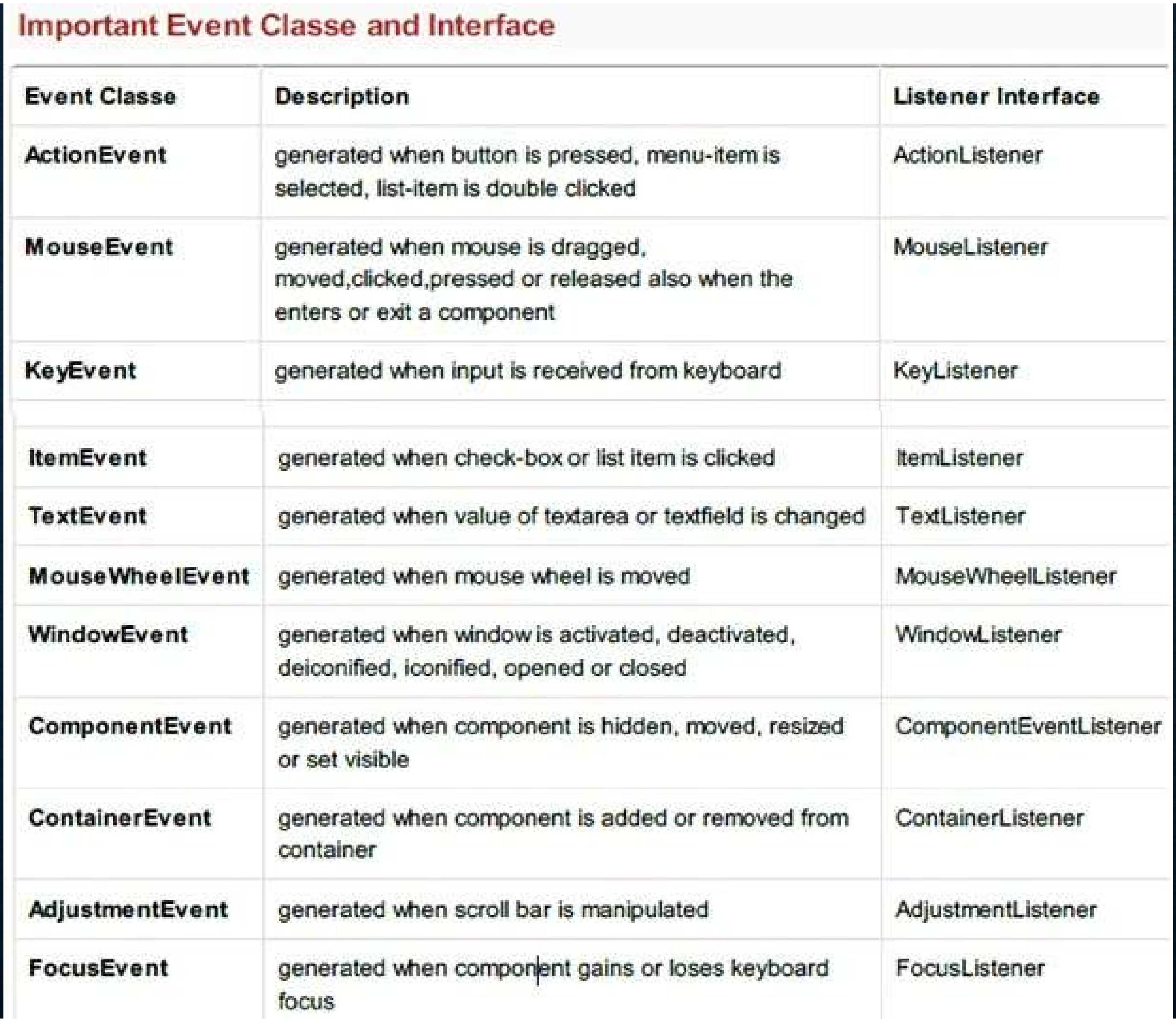
1) The transmission rate of data over a network is much slower than the rate at which the computer can process it
2) Local file system resources can be read and written at a much slower rate than can be processed by the CPU
3) Of course, user input is much slower than the computer



DJS compute powered by aws

AWT


Object
Component
Button
Label
CheckBox
Choice
List
Container
Window
Panel
Frame
Dialog
Applet



Important Event Classe and Interface

Event Classe	Description	Listener Interface
ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released also when the enters or exit a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener
ItemEvent	generated when check-box or list item is clicked	ItemListener
TextEvent	generated when value of textarea or textfield is changed	TextListener
MouseWheelEvent	generated when mouse wheel is moved	MouseWheelListener
WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener
ComponentEvent	generated when component is hidden, moved, resized or set visible	ComponentEventListener
ContainerEvent	generated when component is added or removed from container	ContainerListener
AdjustmentEvent	generated when scroll bar is manipulated	AdjustmentListener
FocusEvent	generated when component gains or loses keyboard focus	FocusListener

CONCLUSION

Congratulations! You have unlocked the power of **Object- Oriented Programming** in **Java**. Embrace the principles of OOP, practice and experiment with code, and watch your programs become more **robust**, **modular**, and **scalable**. Keep coding and continue mastering OOP!

Notes will be uploaded to GitHub, go follow our instagram to find all relevant links

THANKS!

BY DJS COMPUTE