## Strings

- String is a sequence of characters
- Strings are immutable (Cannot be changed)
- Two strings can be combined by using a + operator

Lets look at the below strings:

String s1 = “Learning Java”

String s2 = “Basics”

String s3 = s1 + s2;

## String Methods

String s1 = “Learning Java”

- s1.length() → Returns the length of string i.e. 12
- s1.toUpperCase() →  Converts the string to upper case i.e. LEARNING JAVA
- s1.toLowerCase() → Converts the string to lower case i.e. learning java
- s1.indexOf(“a”) → Returns the position of specified characters in a string i.e. 2
- s1.charAt(5) → Returns the character at specified index i.e. i
- s1.equals(“Python”) → Compares two strings i.e. false
- s1.trim() -> Removes white space from both ends of a string i.e. LearningJava

QASCRIPT

# Arrays

- Array is a container object that can hold a fixed number of values of a single type
- Length of the array can be defined during declaration and it will remain fixed
- Each item in an array is called an element and each element can be accessed by its index

Syntax:

```
int array[] = {2,4,6,8,10};
```

## Methods in Array

- sort() – Sorts the specified array in ascending order
- toString() – Returns a string representation of the contents of the specified array
- equals() – Returns true if two specified arrays of ints are equal to one another
- copyOf() – Copies the specified array
- length() – Returns the length of the specified array

# File and I/O

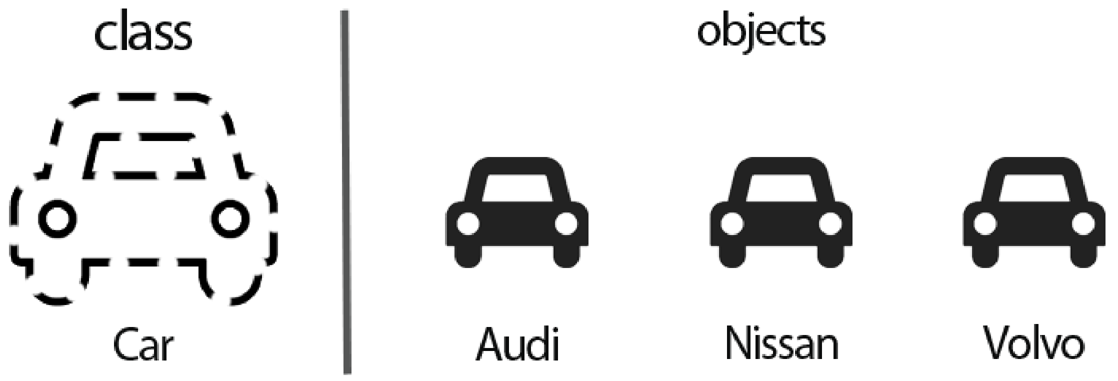
- A stream can be defined as a sequence of data. There are 2 types:
  - InputStream – It is used to read data from a source
  - OutputStream – It is used to write data to a source
- Standard Streams:
  - Standard Input – This is used to accept input from the user keyboard and represented as System.in
  - Standard output – This is used to output the data to the console and represented as System.out
  - Standard Error – This is used to output the error to the console and represented as System.err

- FileInputStream – It is used to read data from the files.
  - Ex – FileInputStream fs = new FileInputStream("C:/demo.txt");

- FileOutputStream – It is used to create a file and write data into it.
  - Ex – FileOutputStream fs = new FileOutputStream("C:/test.txt");

## Classes

- Class is a user-defined blueprint from which individual objects are created.
- Consider a blue print for Car. The car has properties such as weight and color, and methods such as driver and brake
- Every car will have these common properties but the values will be different



## Objects

- Object is an instance of a class
- All instances share the same attributes and behavior of class but values of the attributes are unique
- Just like we construct buildings from blueprints, a class can be used to create an object in code
- Car class defines  weight, color as attributes. We can create 3 objects from class

  Object 1 – Audi: weight = 1500, color = red

  Object 2 – Nissan: weight = 1200, color = black

  Object 3 – Volvo: weight = 1700, color = white

## Constructors

- Constructor initializes an object when it is created
- It has the same name as the class and is very similar to a method except they don’t have return type
- If no constructor is defined for a class, then a default constructor is automatically created by Java
- Constructors are used to assign values to class variables at the time of object creation
- Constructor cannot be abstract, final or static
- Two types of Constructors:
  - Default Constructor – It has no parameters
  - Parameterized Constructor – It has parameters

## Access Modifiers

- They help to restrict the scope of a class, constructor, variable, method or data member
- Four types of access modifiers:
  1. Default – When no access modifier is specified for class/method/members then it has default access modifier. These can be only accessed within the same package
  2. Private – Methods/data members declared as private are only accessible within the same class. Any other class within the same package cannot access these members
  3. Protected – Methods/data members declared as protected are only accessible within the same package or subclasses in different package
  4. Public – Methods/data members declared as public can be accessed from everywhere in the program. No restriction on the public data members

# Non Access Modifiers

- Static Modifier
  - Static keyword is a non-access modifier which can be used with variables, methods and blocks
  - Static member is shared across all instances of the class
  - Static methods belong to a class instead of the object, they can be called without creating the object of the class
  - Static keyword is mainly used for memory management
  - Static variables gets memory only once in the class area when it is loaded
- Final Modifier
  - Final keyword is a non-access modifier used for classes, methods and variables
  - Whenever a variables is declared final, it's value cannot be changed which means it is a constant
  - When a class is declared final, it cannot be inherited
  - When a method is declared final, it cannot be overridden by a derived class

## OOPs

- OOPs is also known as Object Oriented Programming System
- It is based on the concept of objects which can contain methods, variables and attributes
- It has four basic concepts
    - Encapsulation
    - Inheritance
    - Polymorphism
    - Abstraction
- Makes the code easier to maintain, modify and debug
- Many programming languages support OOPs like C++, Java, Python

QS
QASCRIPT

## Encapsulation

- It is the process of wrapping up data and code under a single unit
- Example – A capsule which is made of several medicine compositions
- Ability of an object to hide its data and methods
- Class helps us to encapsulate the fields which hold the state of object and methods which define the actions of the object
- It is a protective shield that prevents the data from being accessed by the code outside the shield
- Encapsulation can be achieved by declaring all variables in the class as private and writing public methods in the class to set and get the values of the variables
- It is a way to achieve data hiding in Java

QASCRIPT

## Abstraction

- Process of displaying essential information and hiding the implementation details
- Abstraction helps to focus on what the object does and not how it does it
- Abstraction is achieved by interfaces and abstract classes
- 100% Abstraction can be achieved by interfaces
- Abstract class can contain both abstract and non-abstract methods
- In Interface, methods are public abstract by default
- Abstract methods are declared without any implementation

# Inheritance In Java

- Inheritance is the ability of a class to inherit the properties and methods of another class
- Class whose features are inherited is known as superclass/base class/parent class
- Class that inherits the other class is known as subclass/derived class/child class
- Java doesn't support multiple inheritance with classes
- Inheritance increases code reusability

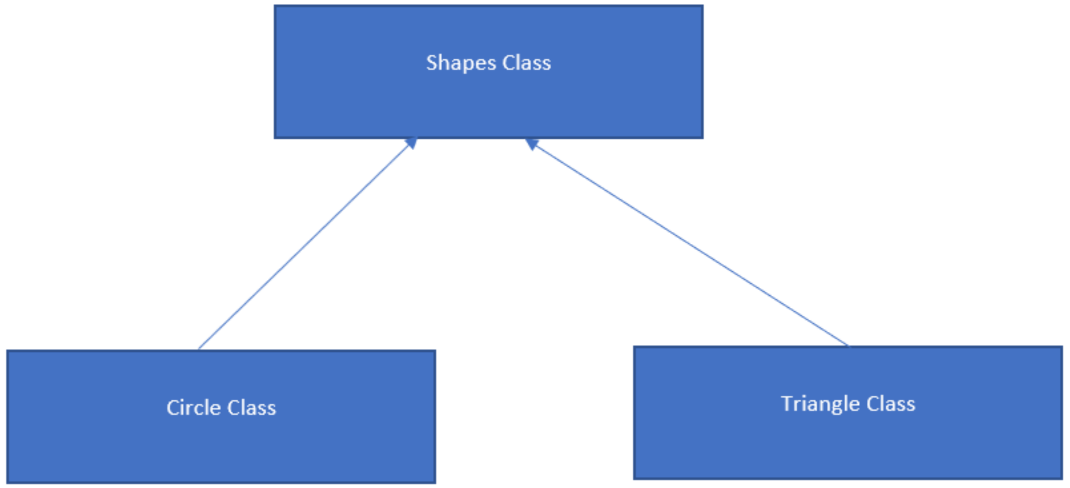
Example: Consider a Shapes class

Triangle class extends properties of Shapes cl

Circle class extends properties of Shapes class

Shapes class is the parent class

Triangle and Circle class are derived classes



QASCRIPT

## Polymorphism

- Ability of different objects to behave differently
- Poly means many and morphs means forms
- 2 types of polymorphism:
    - Compile time Polymorphism – It is a process by which a call to a method is resolved by checking the method signature at compile time. This can be achieved by Method Overloading
    - Run time Polymorphism – It is a process by which a call to overridden method is resolved at run time. It can be achieved by Method Overriding
- In method overloading, multiple methods with same name can exist with different parameters
- In method overriding, method from the derived class can override a method of the base class

## Abstract Class

- Abstract class is used when we want to share code among several closely related classes
- Abstract class can have both abstract and non abstract methods
- Abstract class cannot be instantiated
- If the class is declared as abstract then the sole purpose is for the class to be extended
- A class cannot be both abstract and final
- Abstract class can have constructors and static methods
- Abstract class can be extended using “extends”
- Any class that extends an abstract class must implement all the abstract methods of the super class
- Abstract class doesn’t support multiple inheritance

Syntax:

```
Abstract class Bike{
  abstract void run();
}
```

## Interface

- Interface is an abstract type that is used to specify a behavior that classes must implement
- Interfaces are used to achieve complete abstraction
- Interfaces contains abstract methods which have no implementation
- Interfaces can be used to achieve multiple inheritance in Java
- Variables in interface are final, static and public
- Interfaces cannot be instantiated
- Interfaces can be implemented using "implements"

Example: Selenium WebDriver is an interface

Syntax:

```
interface car{
  void start();
}
```