GLS UNIVERSITY

SEM – IV
0301401 - CORE JAVA

Dr. Disha Shah
Prof. Vidhi Thakkar

Unit – 2

Java Programming Constructs, Classes, Vectors and Array

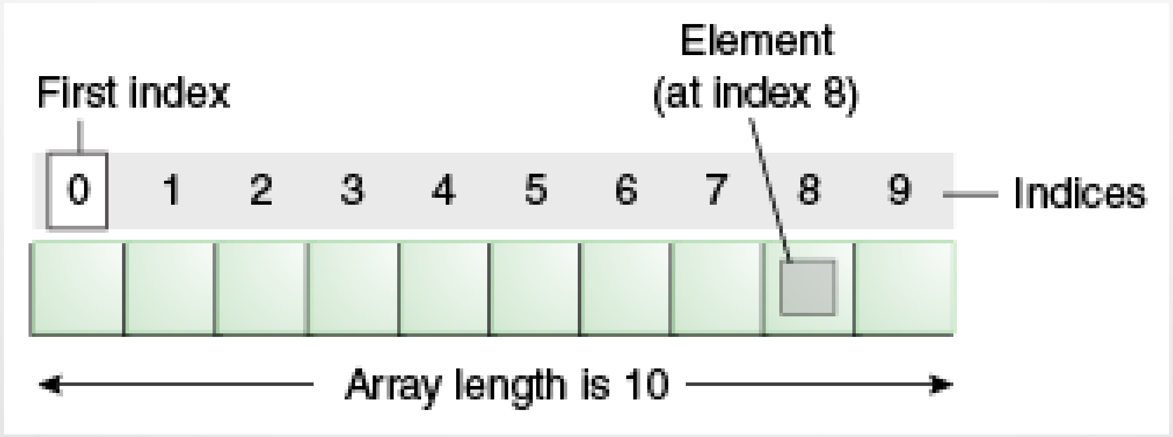
Index

Arrays

Normally, Array is a collection of similar type of elements that have contiguous memory location.

Java array is an object the contains elements of similar data type. It is a data structure where we store similar elements. We can store only fixed set of elements in a java array.

Array in java is index based, first element of the array is stored at 0 index.



Arrays

Advantage of Java Array

- Code Optimization: It makes the code optimized, we can retrieve or sort the data easily.
- Random access: We can get any data located at any index position.

Disadvantage of Java Array

- Size Limit: We can store only fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in java.

Single Dimensional Arrays

- There are two types of array
  - Single Dimensional Array
  - Multi Dimensional Array

- **Single Dimensional Array in java**
- **Syntax:**

```
dataType[] arr; (or)
dataType []arr; (or)
dataType arr[];
```

Single Dimensional Arrays

- **Instantiation**

  arrayRefVar=new datatype[size];

- int a[]={33,3,4,5};//declaration, instantiation and initialization

Multi Dimensional Arrays

- In such case, data is stored in row and column based index (also known as matrix form).
- **Syntax**

  dataType[][] arrayRefVar; (or)

  dataType [][]arrayRefVar; (or)

  dataType arrayRefVar[][]; (or)

  dataType []arrayRefVar[];
- **Example**

  int[][] arr=new int[3][3];//3 row and 3 column

The foreach Loop

- JDK 1.5 introduced a new for loop known as foreach loop or enhanced for loop, which enables you to traverse the complete array sequentially without using an index variable.

Advantage of for-each loop:

- It makes the code more readable.
- It eliminates the possibility of programming errors.

Syntax of for-each loop:

```
for(data_type variable : array | collection){}
```

The foreach Loop

```
class ForEachExample1
{
 public static void main(String args[])
{
   int arr[]={12,13,14,44};

   for(int i:arr)
   {   System.out.println(i);
   }
  }
}
```

Passing Arrays to Methods

Just as you can pass primitive type values to methods, you can also pass arrays to methods. For example, the following method displays the elements in an int array −

Example:

```
public static void printArray(int[] array) {
   for (int i = 0; i < array.length; i++) {
      System.out.print(array[i] + " ");
   }
}
```

Returning Arrays from Methods

- A method may also return an array. For example, the following method returns an array that is the reversal of another array −

Example:

```
public static int[] reverse(int[] list)
{
   int[] result = new int[10];
   for (int i = 0, j = result.length - 1; i < list.length; i++, j--) {
      result[j] = list[i];
   }
   return result;
}
```

Variable Arguments

- Variable arguments can be used when the number of arguments that you want to pass to a method are not fixed.
- The method can accept different number of arguments of same type whenever they are called.
- Syntax:

```
returntype methodname (datatype...arrayname)
```

Introduction

Java is an Object-Oriented Language. As a language that has the Object Oriented feature, Java supports the following fundamental concepts −

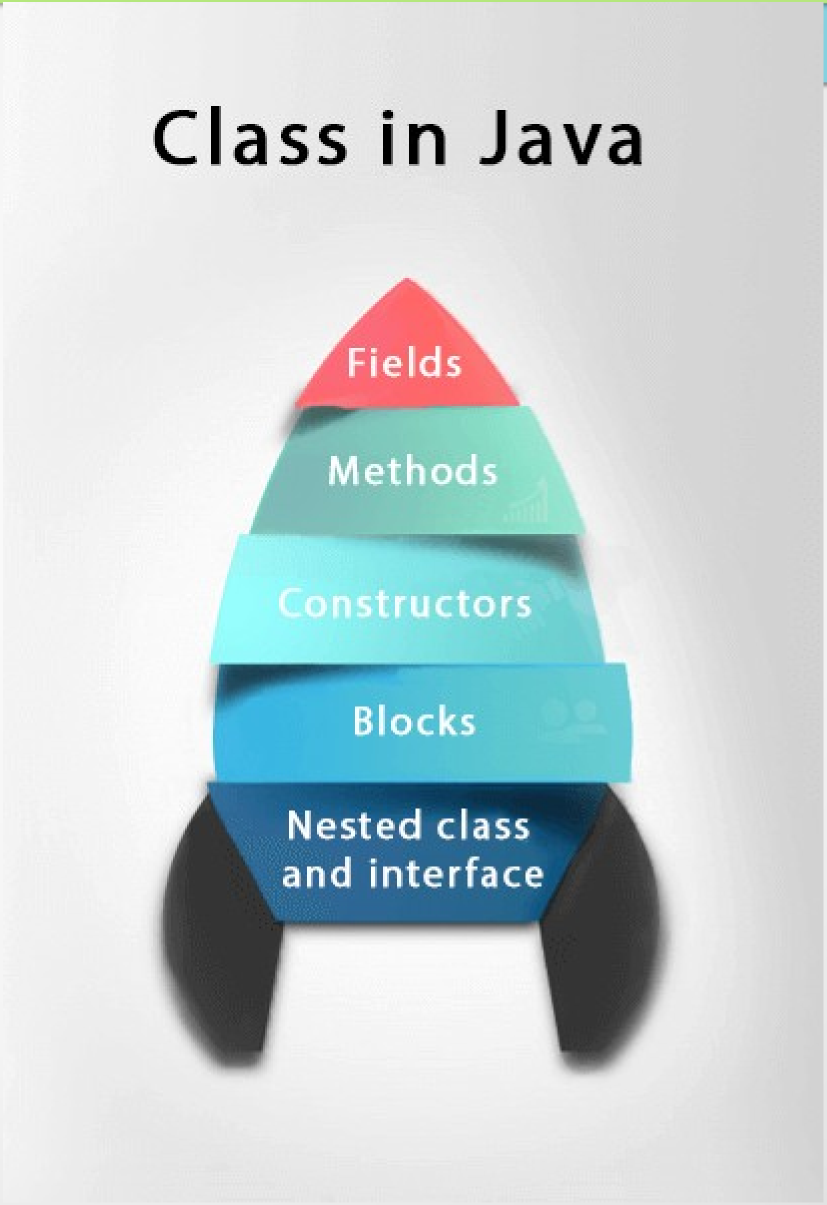
- Polymorphism
- Inheritance
- Encapsulation
- Abstraction
- Classes
- Objects
- Instance
- Method
- Message Parsing

Classes & Objects

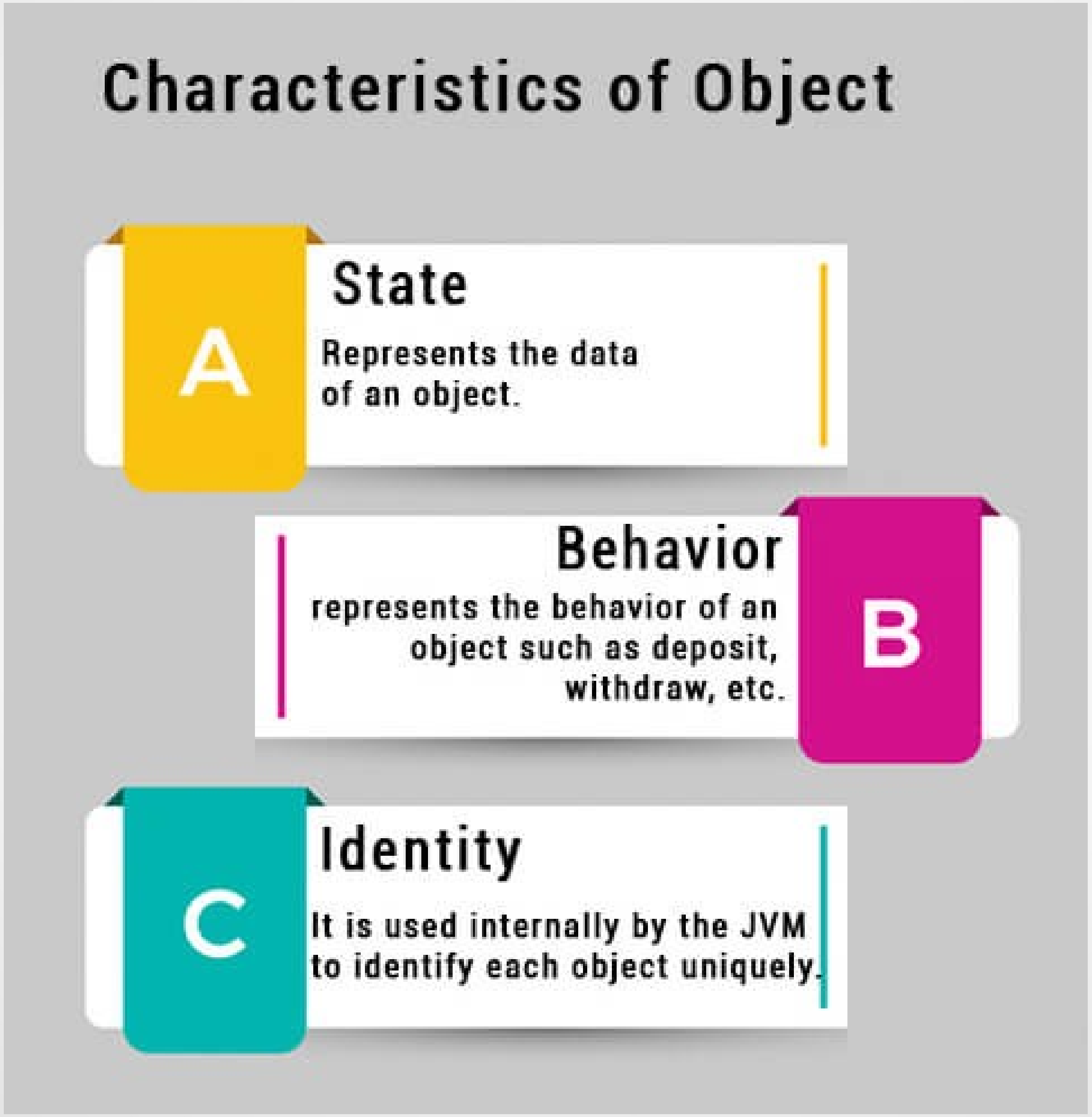
- **Object** - Objects have states and behaviors. Example: A dog has states – color, name, breed as well as behaviors – wagging the tail, barking, eating.
- An object is an instance of a class.
- **Class** - A class can be defined as a template / blueprint that describes the behavior / state that the object of its type support.
- A class is a blueprint or prototype that defines the variables and methods common to all objects of a certain kind,
- We can think of class as a user-defined data types and an object as a variable of that data type that can contain data and methods.

Classes & Objects

- A class may contain:
  - Fields
  - Methods
  - Constructors
  - Blocks
  - Nested class and interface



Classes & Objects



Characteristics of Object

A State

Represents the data of an object.

B Behavior

represents the behavior of an object such as deposit, withdraw, etc.

C Identity

It is used internally by the JVM to identify each object uniquely.

Classes & Objects

```
public class Dog
{
    String breed;
    int age;
    String color;

void barking()
{ }

void hungry()
{ }

void sleeping()
{ }

}
```

Classes & Objects

- There are 2 ways to declare main method.
  - main within the class
  - main outside the class

Classes & Objects

- A class can contain any of the following variable types.
- **Local variables:** Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
- **Instance variables:** Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
- **Class variables:** Class variables are variables declared within a class, outside any method, with the static keyword.
- A class can have any number of methods to access the value of various kinds of methods.
- In the above example, barking(), hungry() and sleeping() are methods.

Classes & Objects

- A class provides the blueprints for objects. So basically, an object is created from a class. In Java, the new keyword is used to create new objects.
- There are three steps when creating an object from a class:
- **Declaration:** A variable declaration with a variable name with an object type.
- **Instantiation:** The 'new' keyword is used to create the object.
- **Initialization:** The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Classes & Objects

```
public class MyClass
 {
  int x = 5;


  public static void main(String[] args)
 {
    MyClass myObj = new MyClass();
    System.out.println(myObj.x);
  }
}
```

Classes & Objects

There are 3 ways to initialize object in Java

- By reference variable
- By method
- By constructor

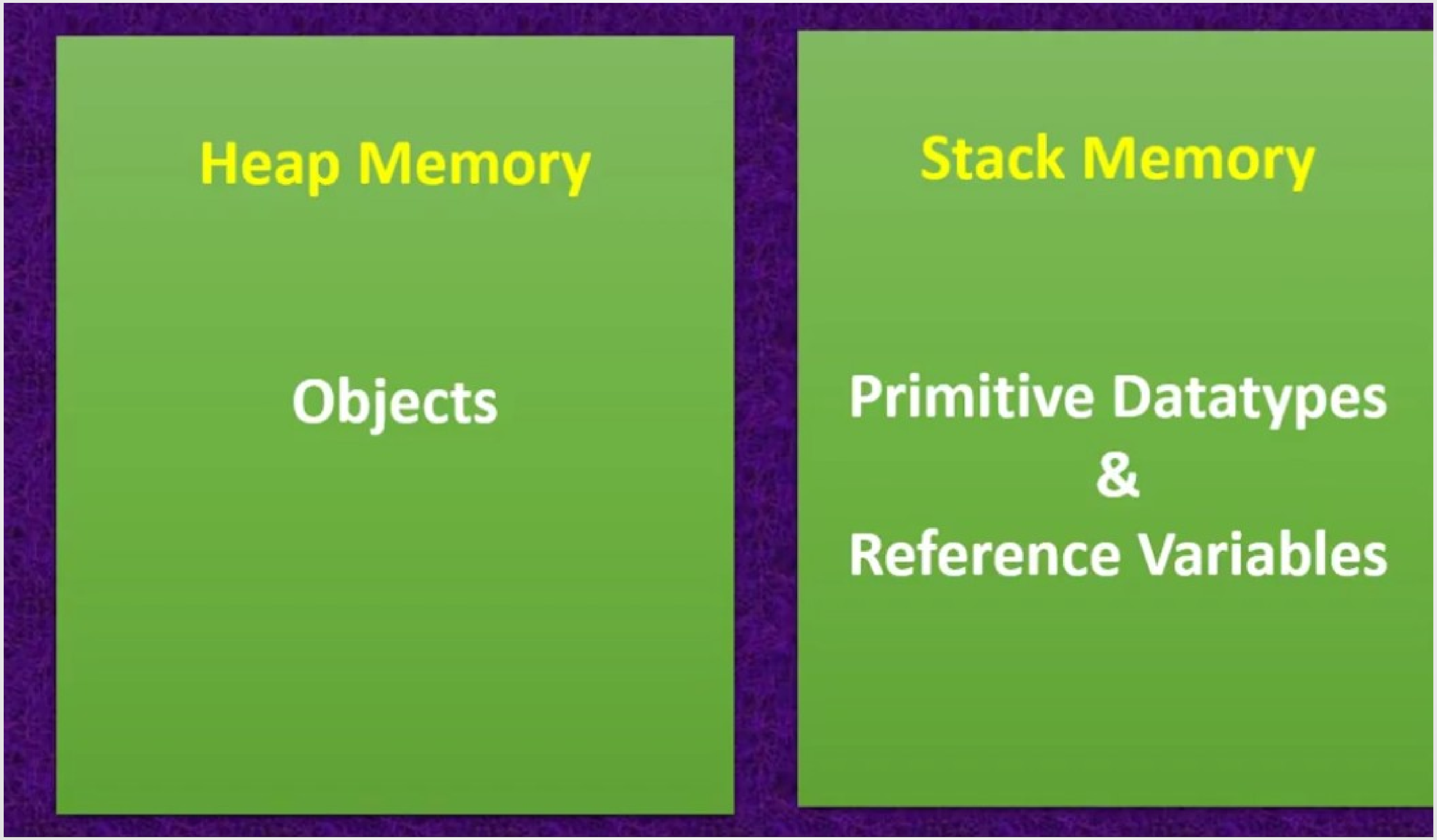
Memory Allocation in Java

- Memory Allocation in Java is the process in which the virtual memory sections are set aside in a program for storing the variables and instances of structures and classes.
- However, the memory isn’t allocated to an object at declaration but only a reference is created.
- For the memory allocation of the object, new() method is used, so the object is always allocated memory on the heap.
- The Java Memory Allocation is divided into following sections :
- Heap
- Stack
- Code
- Static

Memory Allocation in Java

- The code section contains your bytecode.
- The Stack section of memory contains methods, local variables, and reference variables.
- The Heap section contains Objects (may also contain reference variables).
- The Static section contains Static data/methods.

Classes & Objects

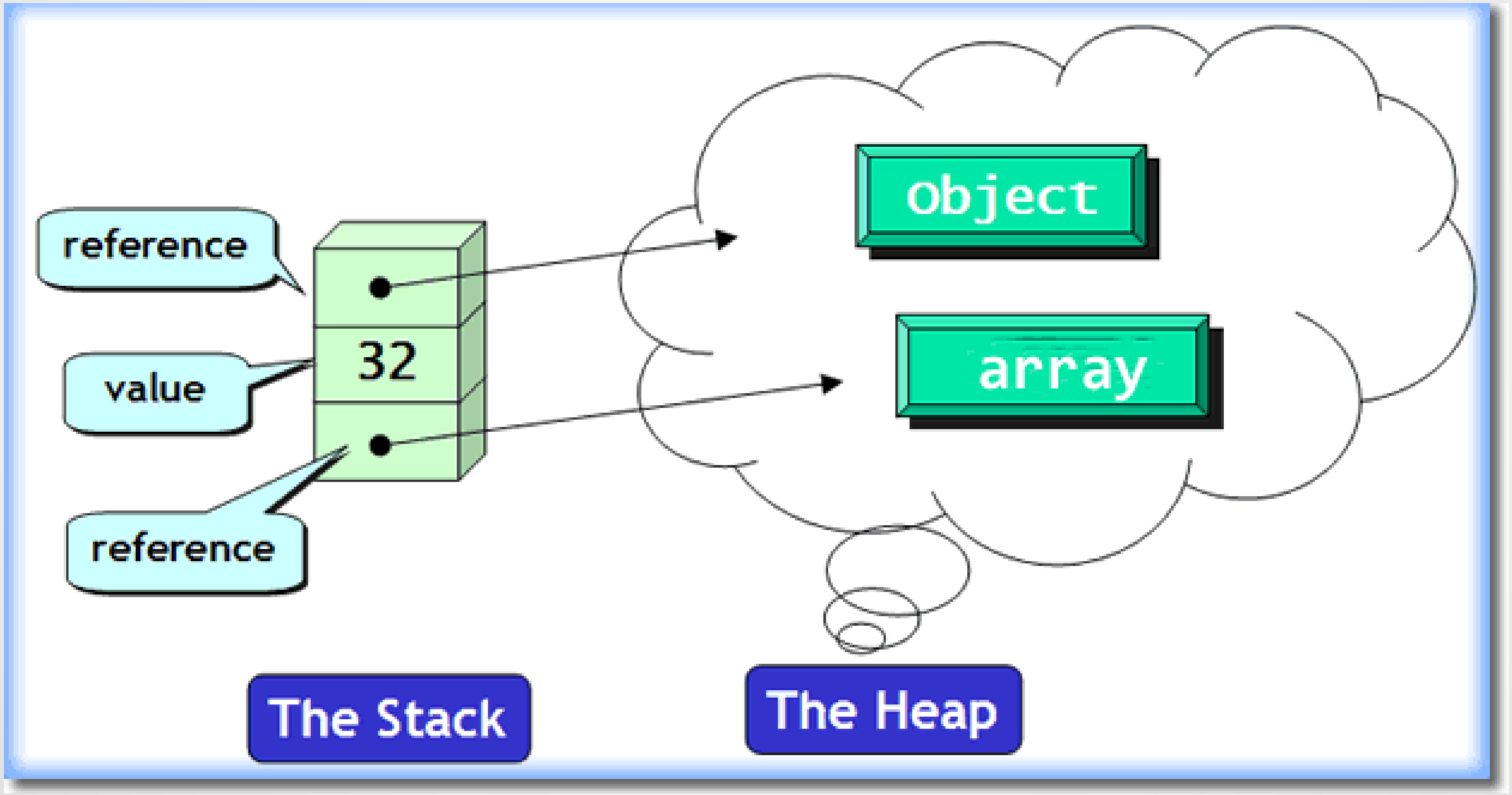


Heap Memory

Objects

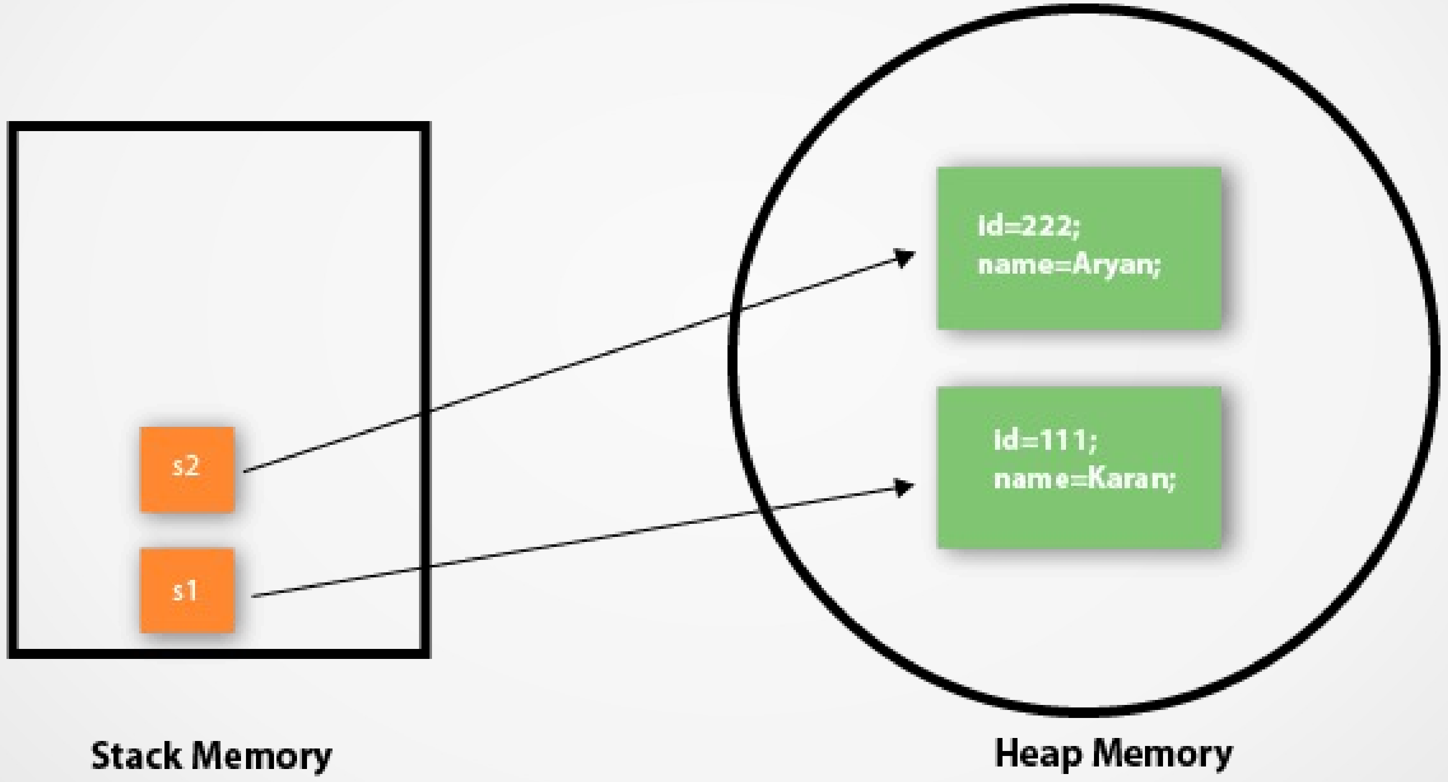
Stack Memory

Primitive Datatypes
&
Reference Variables



reference
value
reference
32
Object
array
The Stack
The Heap

Classes & Objects



Classes & Objects

We can create multiple objects by one type

Rectangle r1=new Rectangle(), r2=new Rectangle();

Methods in Java

- Same as functions in programming language
- None of the methods can be declared outside the class.
- Why use methods?
  - To make code reusable
  - To parameterize the code
  - For top-down programming
  - To simplify the code

Methods in Java

- **Two types of methods**
  - Instance methods
  - Class methods (Static methods)
- Instance methods are used to access/manipulate the instance variables but can also access class variables.
- Class methods can access / manipulate class variables but cannot access instance variable unless and until they use an object for that purpose.

Method Overloading in Java

- If a class have multiple methods by same name but different parameters, it is known as **Method Overloading.**
- If we have to perform only one operation, having same name of the methods increases the readability of the program.
- Advantage of method overloading?
  - Method overloading increases the readability of the program.
- There are **three ways to overload the method in java**
  - By changing number of arguments
  - By changing the data type
  - By changing the sequence

Constructors in Java

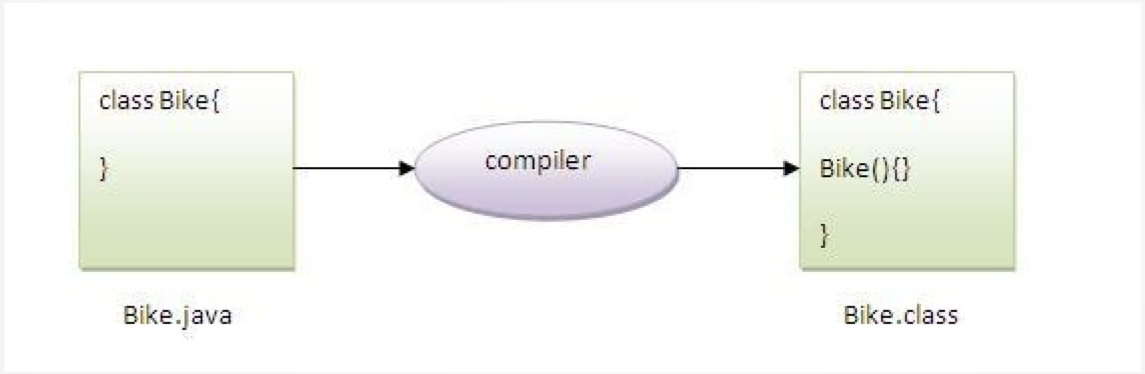
- Constructor in java is a special type of method that is used to initialize the object.
- Java constructor is invoked at the time of object creation.
- It constructs the values i.e. provides data for the object that is why it is known as constructor.
- **Rules for creating java constructor**
  - Constructor name must be same as its class name
  - Constructor must have no explicit return type
- **Types of java constructors**
  - Default constructor (no-arg constructor)
  - Parameterized constructor

Constructor v/s Methods

Constructor	Methods
Do not have any return type not even void.	Will have a return type
Will have the same name as that of class.	Can have any name.
Invoked as soon as the object is created and not thereafter	Invoked after the object is created and can be called any number of times
Constructors cannot be inherited	Methods can be inherited
Constructors can be overloaded	Methods can also be overloaded.
Constructor can be private, public, protected.	Methods can also be private, public, protected.
Role of constructor is to initialize the object	Role of methods is to perform operations
Constructors cannot be abstract, final, static.	Methods can be abstract, final or static.
Constructor is invoked implicitly.	Method is invoked explicitly.
The java compiler provides a default constructor if you don't have any constructor.	Method is not provided by compiler in any case.

Default Constructor

- If there is no constructor in a class, compiler automatically creates a default constructor.



What is the purpose of default constructor?

- Default constructor provides the default values to the object like 0, null etc. depending on the type.

Parameterized Constructor

- A constructor that have parameters is known as **parameterized constructor.**

  Why use parameterized constructor?

- Parameterized constructor is used to provide different values to the distinct objects.

Constructor Overloading

- Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter lists.
- The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

Cleaning up unused Objects

- Objects are allocated memory from the heap memory and when they are not needed their allocated memory should be reclaimed.
- The clean-up code is tedious and often error-prone.
- Java allows programmer to create as many objects as they want, but frees them from deallocating memory objects.
- JRE deletes objects when it determines that it is no longer required.
- It has its own set of algorithms for deciding when the memory must be deallocated.
- This automated process is known as **garbage collection.**

Garbage Collector

- The JRE has a garbage collector that periodically frees the memory used by objects that are no longer used.
- Two approaches used
  - Reference counting
  - Tracing

- **Reference counting** maintains a reference count for every object.
- A newly created object will have count as 1.
- Throughout its lifetime, the object will be referred to by many other object thus incrementing the reference count and as the referencing object move to other objects, the reference count for that particular object is decremented.
- Once the particular object is 0, the object can be garbage collected.

Garbage Collector

- **Tracing technique** traces the entire set of objects and all objects having reference on them are marked in some way.
- Tracing garbage collector algorithm is also known as **mark and sweep collector.**
- It scans Java’s dynamic memory areas for objects, marking those objects that are referenced.
- After all the objects are investigated, the objects that are not marked are assumed by the garbage and their memory is reclaimed.

Garbage Collector

- The garbage collector either synchronously or asynchronously.
- The garbage collector executes synchronously when the system runs out of memory or asynchronously when the system is idle.
- It can be invoked by System.gc() or Runtime.gc()
- There are two ways to do it :
  - **Using System.gc() method** : System class contain static method gc() for requesting JVM to run Garbage Collector.
  - **Using Runtime.getRuntime().gc() method** : Runtime class allows the application to interface with the JVM in which the application is running. Hence by using its gc() method, we can request JVM to run Garbage Collector.

Finalization

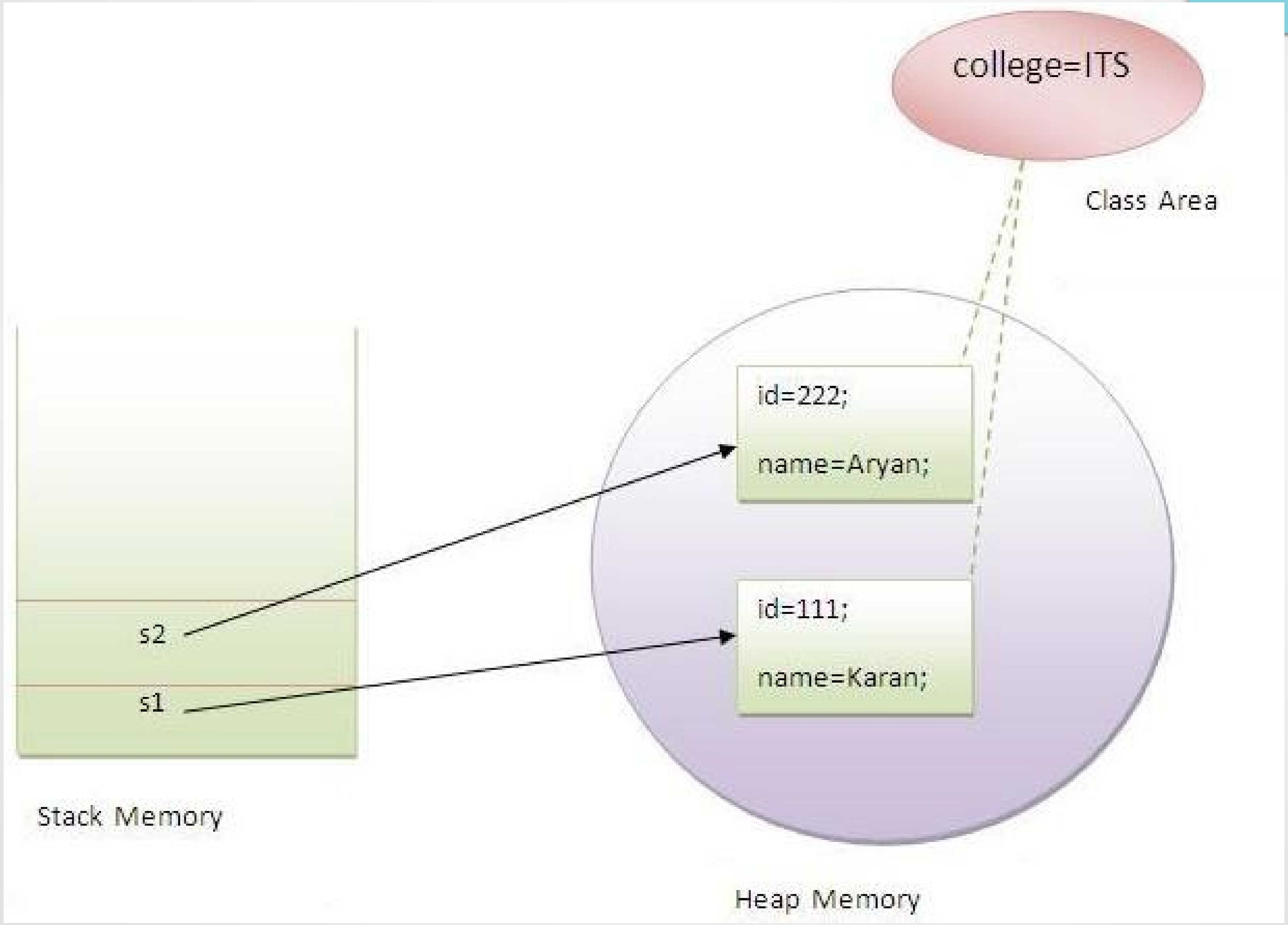
- Before an object gets garbage collected, the garbage collector gives the object an opportunity to clean up itself through a call to the object’s finalize() method.
- This process is known as **finalization.**
- All occupied resources can be freed in the method.
- The finalize() method in java.lang.Object class.

Advantages & Disadvantages

- Advantages:
  - Helps in freeing the programmer from worrying about the deallocation of memory.
  - Helps in ensuring integrity of programs.
- Disadvantages:
  - Overhead to keep track of which objects are being referenced by the executing program and which are not being referenced.
  - Overhead on finalization and freeing memory of the unreferenced objects.
  - So, this will incur more CPU time.

Static Variables

- If you declare any variable as static, it is known static variable.
- The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees,college name of students etc.
- The static variable gets memory only once in class area at the time of class loading.
- It makes your program memory efficient (i.e it saves memory).
- Suppose there are 500 students in my college, now all instance data members will get memory each time when object is created. All student have its unique rollno and name so instance data member is good. Here, college name refers to the common property of all objects. If we make it static,this field will get memory only once.
- Java static property is shared to all objects.


college=ITS
Class Area
id=222;
name=Aryan;
id=111;
name=Karan;
s2
s1
Stack Memory
Heap Memory

Static Method

- If you apply static keyword with any method, it is known as static method.
- A static method belongs to the class rather than object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- Static method can access static data member and can change the value of it.

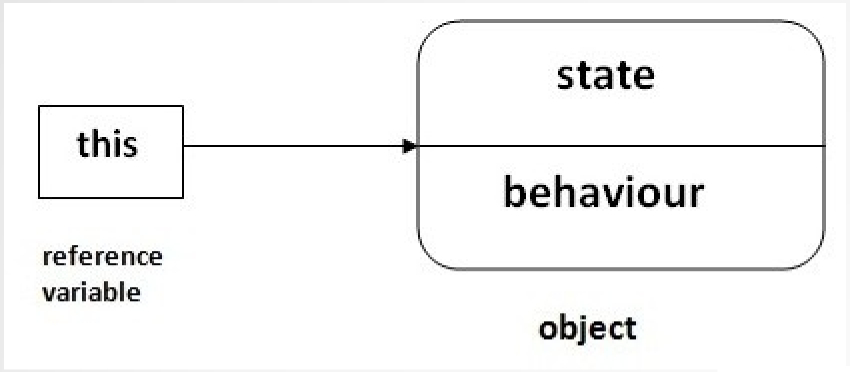
Restrictions for static method

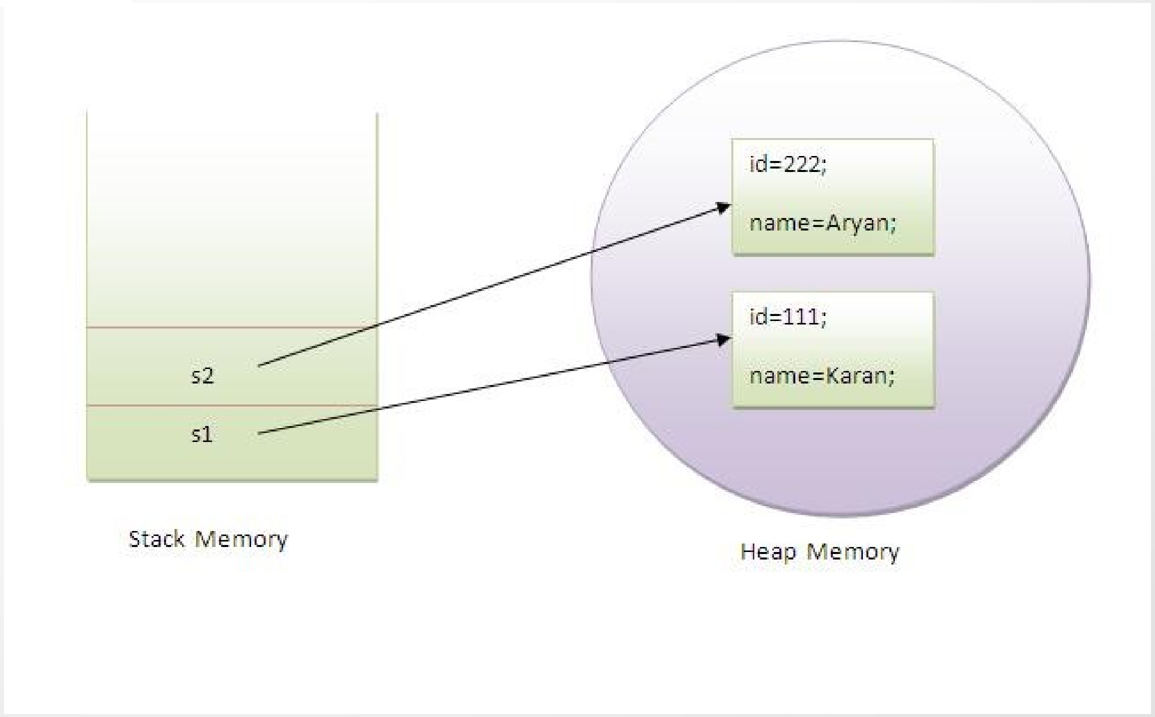
- The static method can not use non static data member or call non-static method directly.
- this and super cannot be used in static context.

this Method

- In java, this is a reference variable that refers to the current object.
- this keyword can be used to refer current class instance variable.
- this() can be used to invoke current class constructor.
- this keyword can be used to invoke current class method (implicitly)
- this can be passed as an argument in the method call.
- this can be passed as argument in the constructor call.
- this keyword can also be used to return the current class instance.

this Method


this
reference variable
state
behaviour
object


s2
s1
Stack Memory
id=222;
name=Aryan;
id=111;
name=Karan;
Heap Memory

Command line arguments

- The java command-line argument is an argument i.e. passed at the time of running the java program.
- The arguments passed from the console can be received in the java program and it can be used as an input.
- So, it provides a convenient way to check the behavior of the program for the different values.
- You can pass N (1,2,3 and so on) numbers of arguments from the command prompt.

For-each loop

- It starts with the keyword for like a normal for-loop.
- Instead of declaring and initializing a loop counter variable, you declare a variable that is the same type as the base type of the array, followed by a colon, which is then followed by the array name.
- In the loop body, you can use the loop variable you created rather than using an indexed array element.
- It's commonly used to iterate over an array or a Collections class (eg, ArrayList)
- It makes the code more readable.
- It eliminates the possibility of programming errors.

For-each loop

Syntax:

```
for (type var : array)
{
    statements using var;
}
```

is equivalent to:

```
for (int i=0; i<arr.length; i++)
{
    type var = arr[i];
    statements using var;
}
```

For-each loop

Example:

```
 for(int i:arr)
{
    System.out.println(i);
}
```

Limitations of for-each loop

For-each loops are not appropriate when you want to modify the array:

```
for (int num : marks)
{
    // only changes num, not the array element
    num = num*2;   }
```

For-each loops do not keep track of index. So we can not obtain array index using For-Each loop

```
for (int num : numbers)
{
    if (num == target)
    {
        return ???;   // do not know the index of num
    }}
```

Limitations of for-each loop

For-each only iterates forward over the array in single steps

```
// cannot be converted to a for-each loop
for (int i=numbers.length-1; i>0; i--)
{
      System.out.println(numbers[i]);}
```

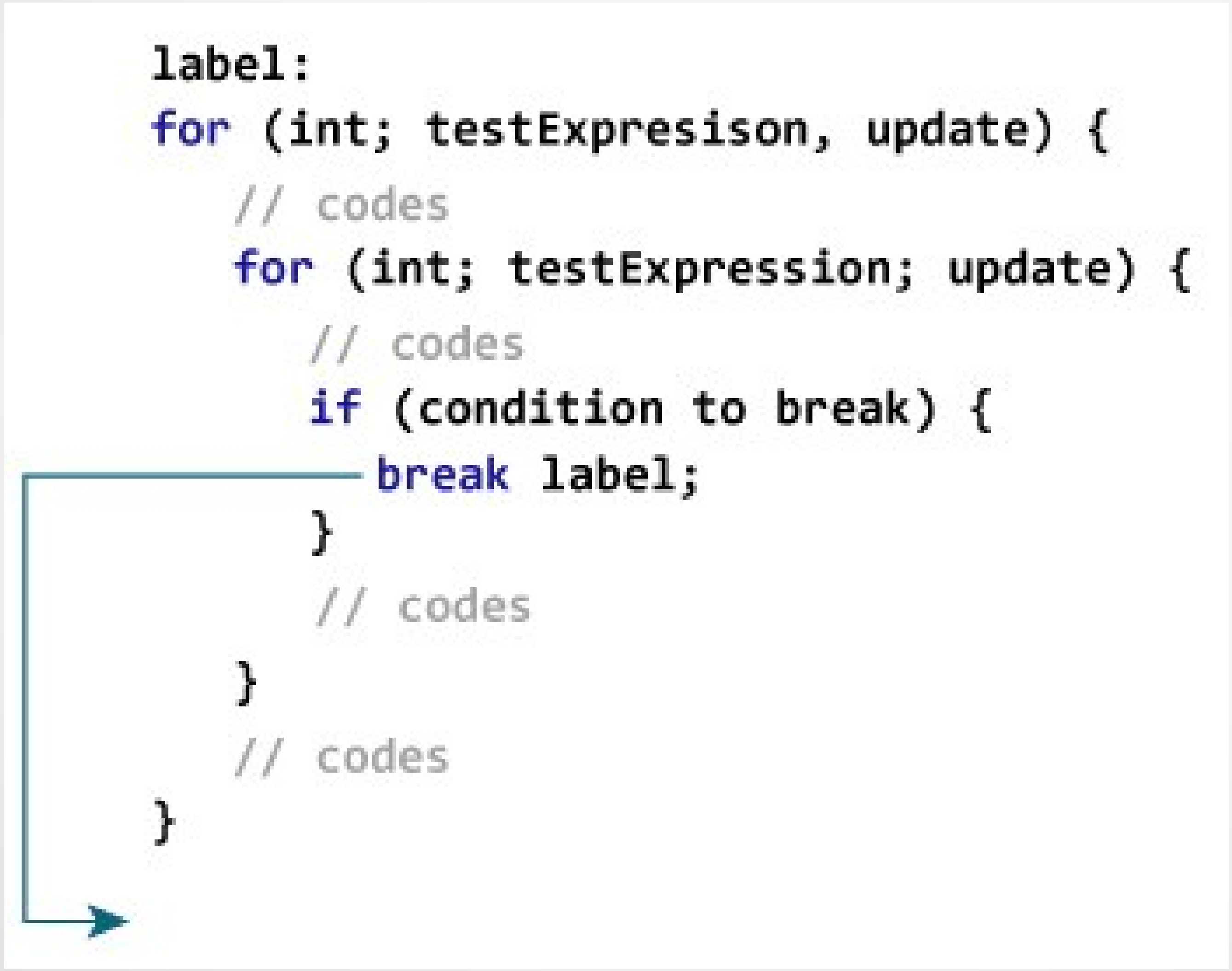
For-each cannot process two decision making statements at once

```
// cannot be easily converted to a for-each loop
for (int i=0; i<numbers.length; i++)
{
    if (numbers[i] == arr[i])
    { ...
    } }
```

Breaking mechanism

- Break
- Continue
- Return

Labeled break statement



```
label:
for (int; testExpresison, update) {
    // codes
    for (int; testExpression; update) {
        // codes
        if (condition to break) {
            break label;
        }
        // codes
    }
    // codes
}
```

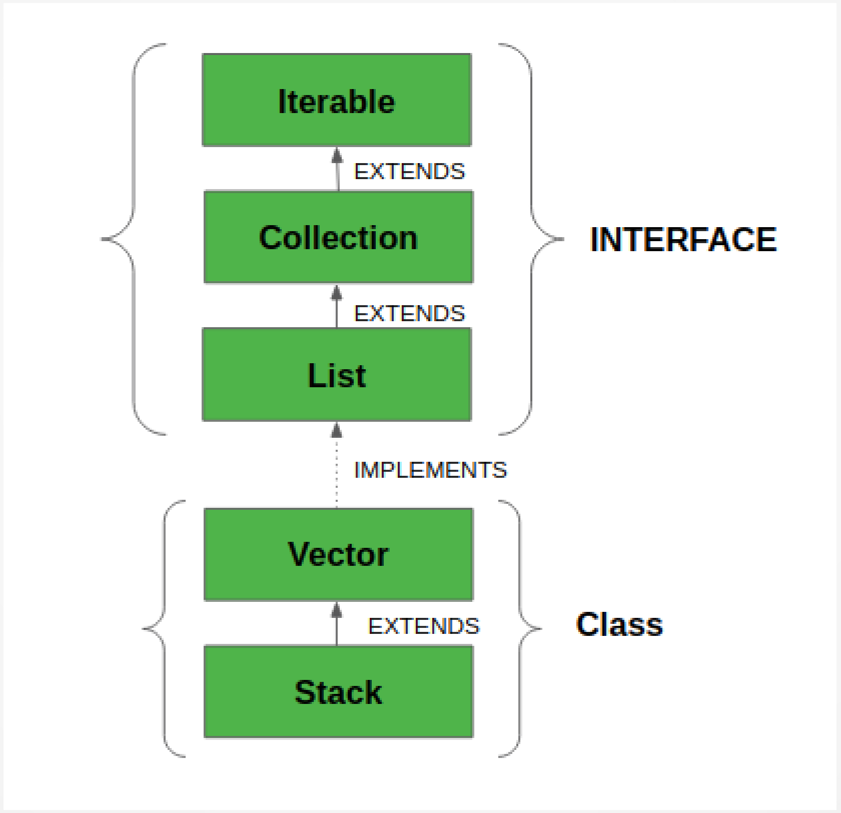
Labeled break statement

```
class LabeledBreak {
   public static void main(String[] args) {

      first:
      for( int i = 1; i < 5; i++) {
         second:
         for(int j = 1; j < 3; j ++ ) {
            System.out.println("i = " + i + "; j = " +j);

            if ( i == 2)
            break first;
         }
      }
   }
}
```

Vector Class

- Vector implements a dynamic array.
- It is similar to ArrayList.
- A vector is similar to a dynamic array in that its size can be increased or decreased.
- Unlike arrays, it has no size limit and can store any number of elements.

Vector Class



Java Vector

- The Vector class implements a growable array of objects.
- Vector is like the dynamic array which can grow or shrink its size.
- Unlike array, we can store n-number of elements in it as there is no size limit.
- It is a part of Java Collection framework since Java 1.2.
- It is found in the java.util package and implements the List interface.

Constructors

- Vector<E> v = new Vector<E>();
- Vector<E> v = new Vector<E>(int size);
- Vector<E> v = new Vector<E>(int size, int incr);
- Vector(Collection c)

Java Vector Methods

- Some of the following are the list of Vector class methods:
  - **add()** It is used to append the specified element in the given vector.
  - **addAll()** It is used to append all of the elements in the specified collection to the end of this Vector.
  - **addElement()** It is used to append the specified component to the end of this vector. It increases the vector size by one.
  - **get()** It is used to get an element at the specified position in the vector.
  - **remove()** It is used to remove the specified element from the vector. If the vector does not contain the element, it is unchanged.
  - **removeAll()** It is used to delete all the elements from the vector that are present in the specified collection.

Java Vector Methods

- **capacity()** It is used to get the current capacity of this vector.
- **removeElementAt()** It is used to delete the component at the specified index.
- **firstElement()** It is used to get the first component of the vector.
- **lastElement()** It is used to get the last component of the vector.