What is a class?

- A Java class can be defined as a template or blueprint which describes state/behavior of it's object.
- In Other Word a class is used to create Objects.

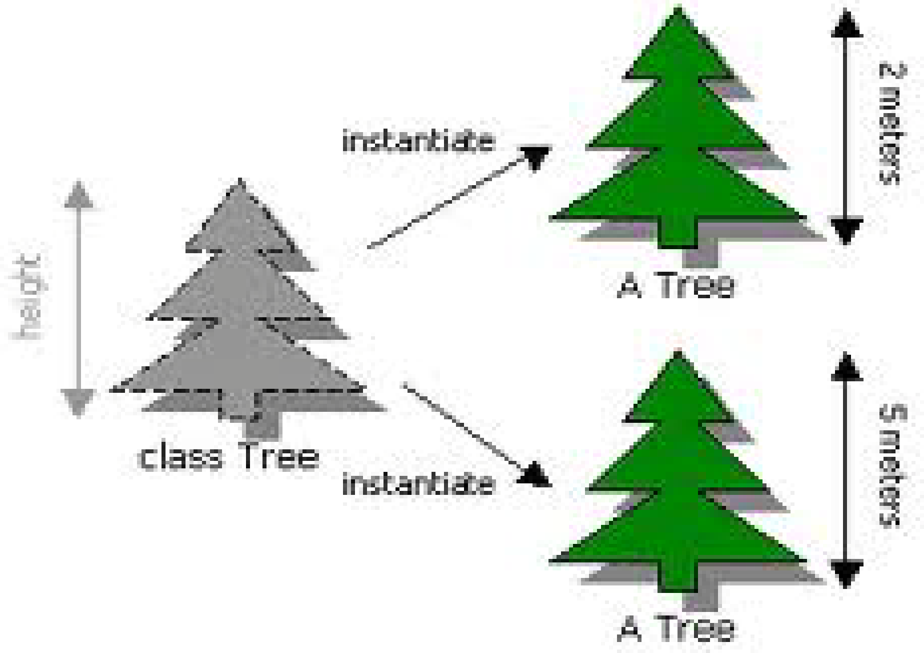


Fig. 1: Instantiating two Trees from the Tree class

Creating a class

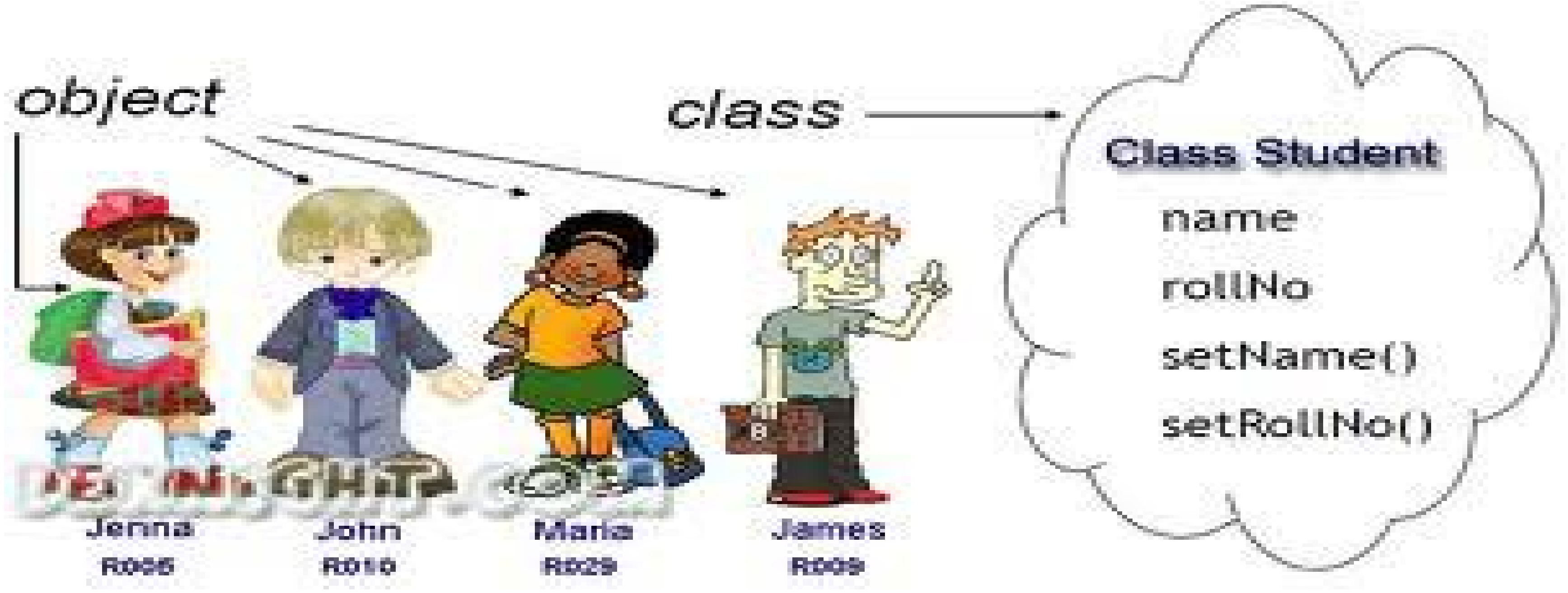
- A class is declared by using the keyword “class”. E.g.
- public class Tree

```
 {
//This is our Tree class
//All codes goes here


 }
```

What are objects?

- Objects are nothing but the instance of the class.
- A single class can create any number of unique objects.



Creating objects

- In java an object is created when someone says “new”.
- At each “new” , a new object of a class is created. E.g.

```
new dog();
new tree();
new student();
```

Where objects lives?

- Objects lives in Java heap.

What is java heap?

Java heap is nothing but the memory space taken by JVM from the OS.

All objects are created in this heap(space). Whenever JVM encounters “new” keyword, it creates an object in heap.

- When there is no space in heap to create objects, then JVM throws “Out of Memory” error.

Creating First Object

- public class Student

```
{
public static void main(String[] args)
{
new Student();
}
}
```

Garbage Collection

- Garbage Collection is the mechanism provided by JVM, to clean out the Heap, so that new objects can be created.
- It destroys the objects which are “not in use” or eligible for “garbage collection”.
- Any object is said to be eligible for garbage collection (GC) if there is no “Reference Variable” attached to it.
- So what is a **“Reference Variable”**??????

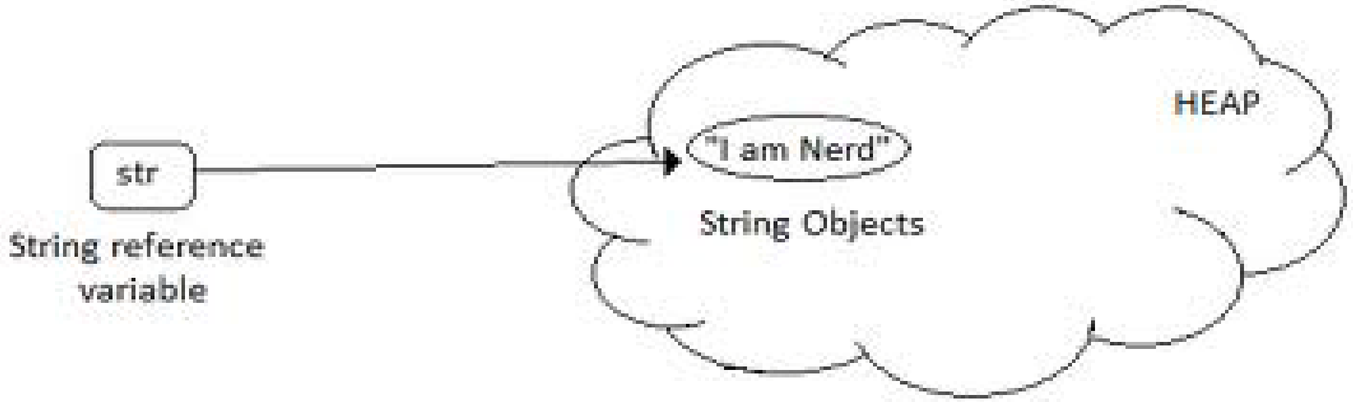
Java Variables and its type

- Variables are named space of memory which stores the **data.**
- There are two types of variables :

➢ Primitive variables

➢ Reference variables

- Reference variables are those variables which stores only address of an “object.



Assigning Objects to Reference Variable

- Before creating a reference variable we have to specify "object of which class" it is going to refer.
- In other word we need to specify the "class type" of the reference variable. E.g.

```
public class Test
{
public  static void main(String[] args)
{
Test t = new Test();
 //where t is the reference variable which stores the
Address of Test object in heap
}
}
```

Primitive and Non primitive datatypes

Primitive Datatype

- Primitive datatypes are defined by the programming language.
- These are

➢ Integer type

➢ Floating type

➢ Character

➢ Boolean

Non-Primitive Datatype

- Non-Primitive (or Reference) datatype are defined by programmer.
- In this the datatype of the variable is the Class whose object it is going to refer.

Test t = new Test();

//where Test is the datatype of t

Declaration and initialization of primitive variables

- **Integer**
  - **int i = 234242425;**
  - **long l = 284798247287427427428947l;**

//suffix 'l' is must otherwise compiler will treat it as integer.

- **Float**
  - **float = 2342.34f**

//suffix 'f' is must otherwise compiler will treat it as a double

- **double = 2984728472424789 27.2942949274**

- **Character**

Char c = 'j'; // only single character is allowed

- **Boolean**

Boolean b = true // only true or false is allowed

Java String and String Concatenation

- In java “String” is a class and not a datatype and it can be instantiated like other classes

String s = new String();

- String Concatenation is basically a way to combine two or more strings into a single string. This is done by using ‘+’ operator.

String s = “We” + “are” + “learning” + “java” + “.”

- String values can be concatenate with any other datatype.

boolean b = true;

String s = “this is” + “ “ + b

Java Operators

Arithmetic operators

- ‘+’ → additive operators/string concatenator
- ‘-’ → subtraction operator
- ‘*’ → multiplication operator
- ‘/’ → division operator
- ‘%’ → remainder operator

Unary operators

- ‘++’ → increment operator
- ‘--‘ → decrement operator
- ‘!’ → logical compliment operator

Java Operator Continues

Equality and Relational

- '==' → Equal to
- '!=' → Not equal to
- '>' → Greater than
- '<' → Less than
- '>=' → Greater than or equal to
- '<=' → Lesser than or equal to

Conditional Operators

- '&&' → Conditional – And
- '||' → Conditional – Or
- '?:' → Ternary Operator

Ternary Operator

- The ternary operator or ?, is a shorthand if else statement. It can be used to evaluate an expression and return one of two operands depending on the result of the expression.

-
  ```
  boolean b = true;
  String s = ( b == true ? "True" : "False" );
  ```

Or the above can be written as

-
  ```
  boolean b = true;
  String s;
  if(b == true){
      s = "True";
  }else{
      s = "False";
  }
  ```