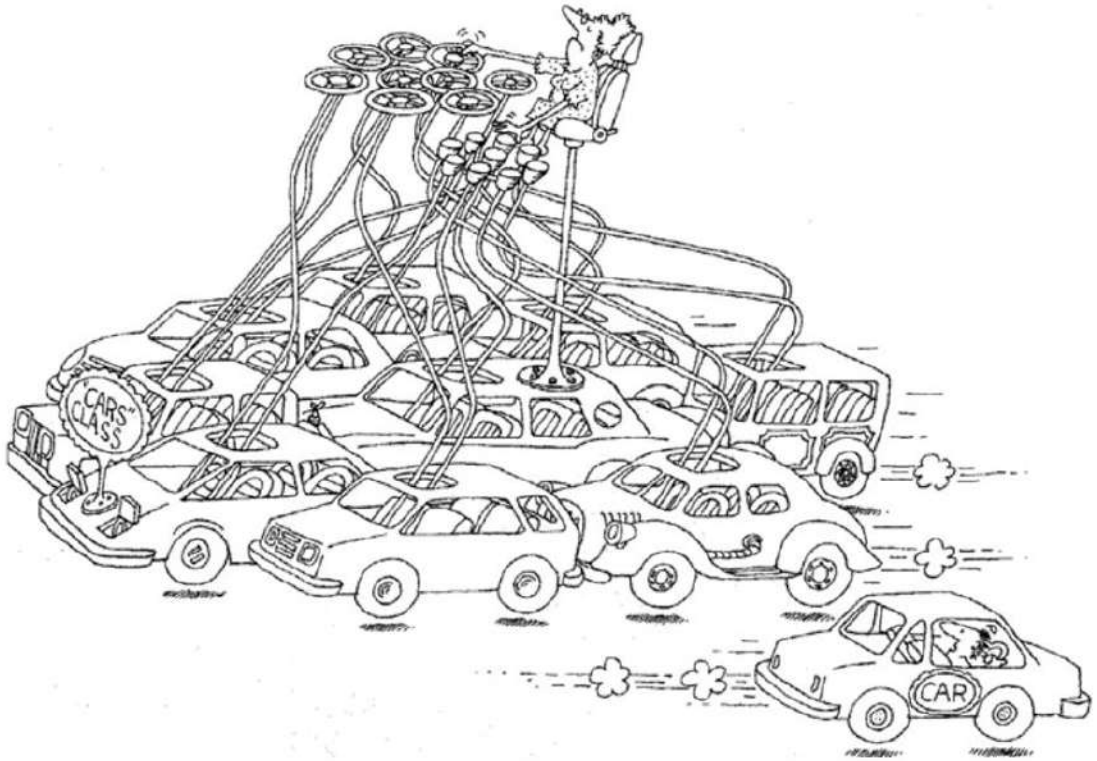
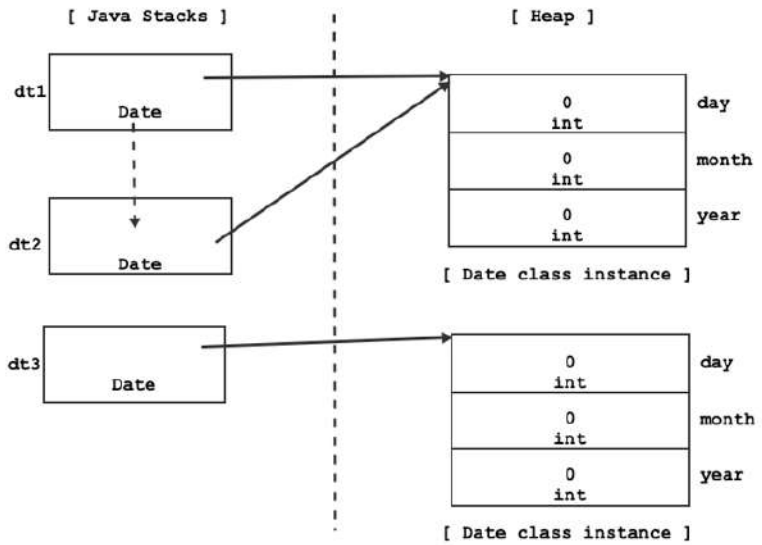
Class

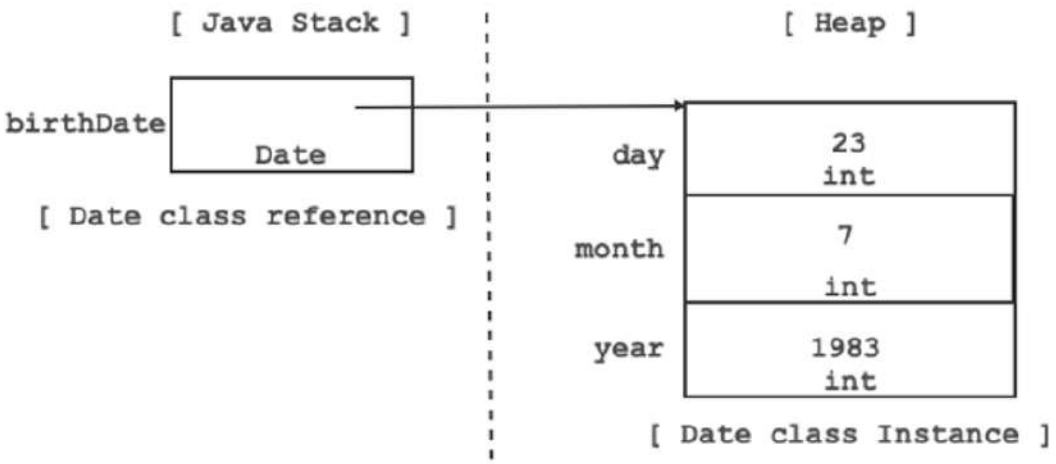


```
Date dt2 = dt1;  //Shallow copy of references

Date dt3 = new Date( );
```

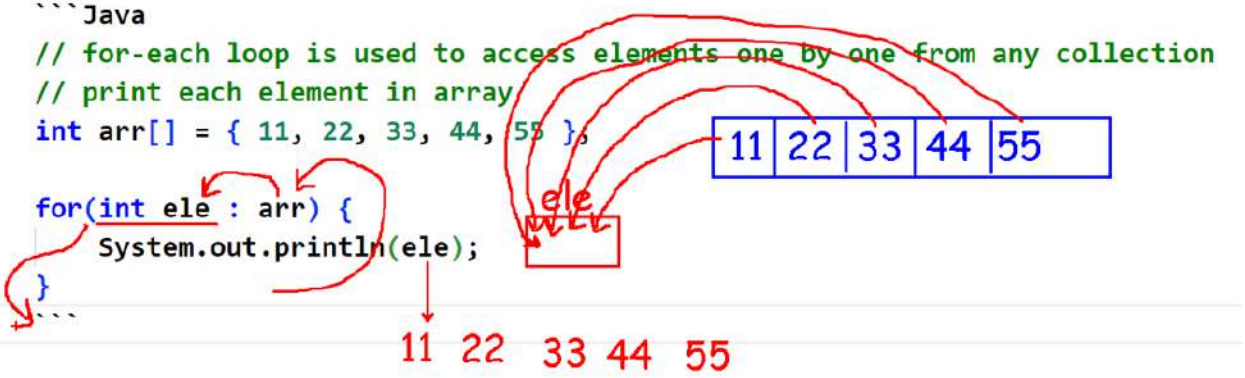



```
Syntax of creating instance/object:
//ClassName refName = new ClassName( );

1. Date dt1 = new Date( );

2. Employee emp = new Employee( );

3. Account acc; //reference
   acc = new Account( );   //Instance
```

```
Date birthDate = new Date( );
```



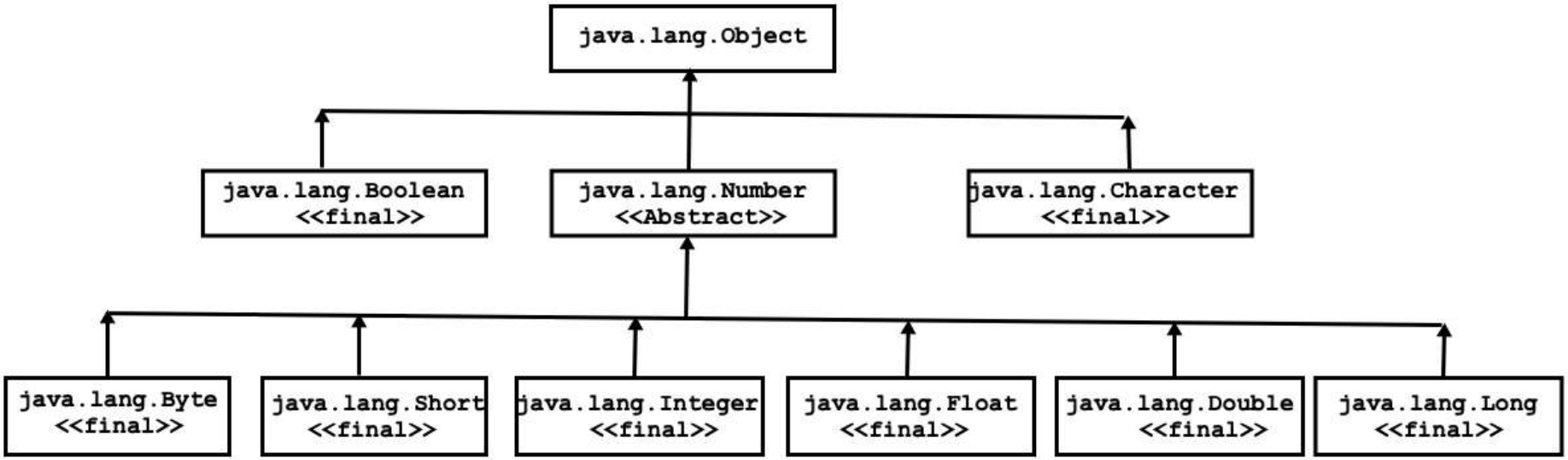
```Java
// for-each loop is used to access elements one by one from any collection
// print each element in array
int arr[] = { 11, 22, 33, 44, 55 };

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



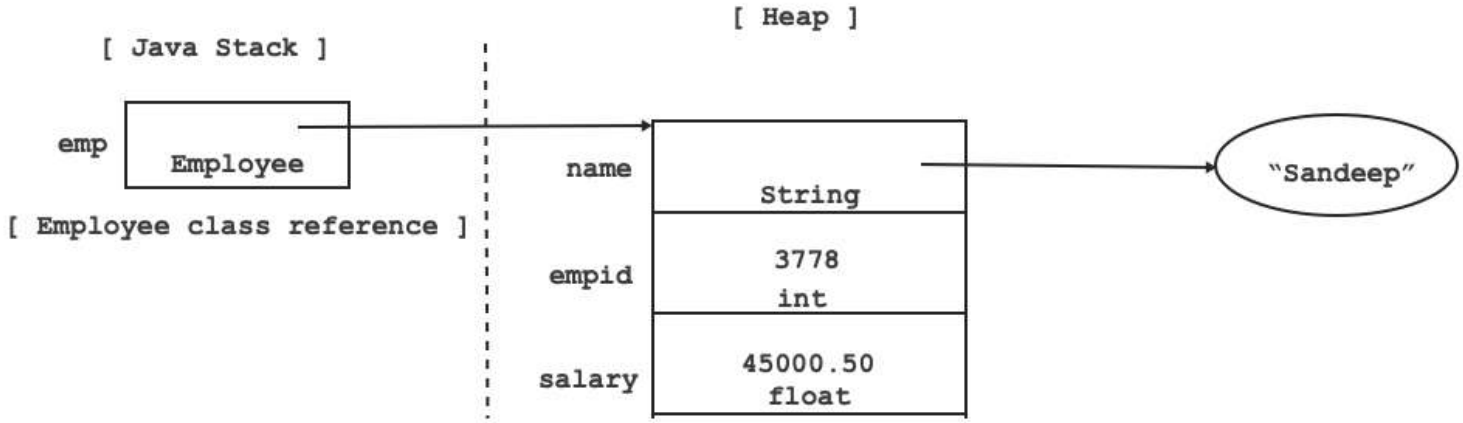
getters/setters provide controlled access of
private fields outside the class.

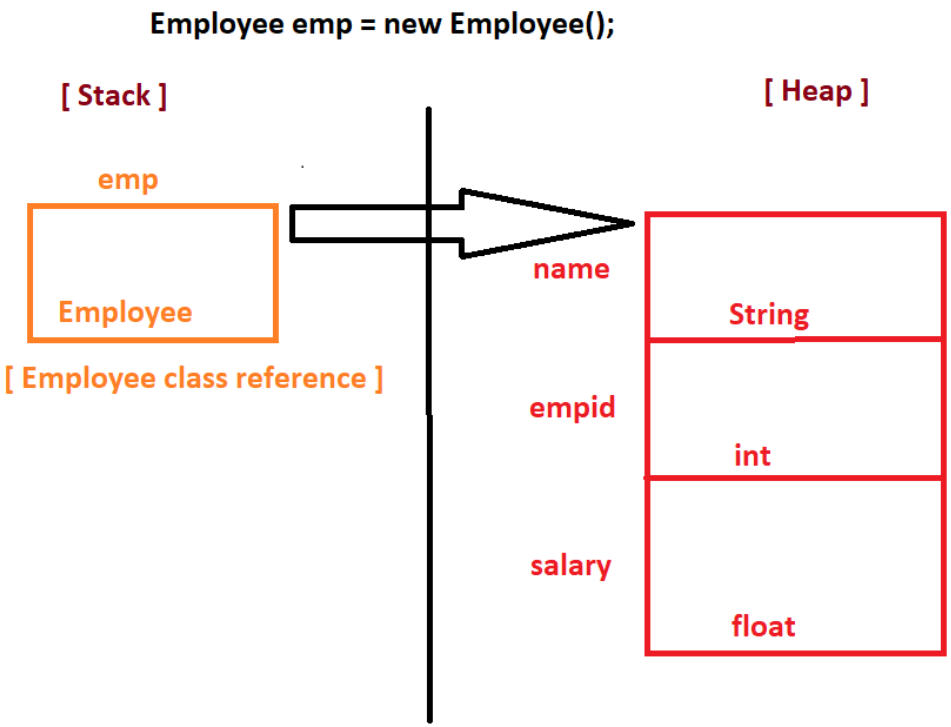
controlled access

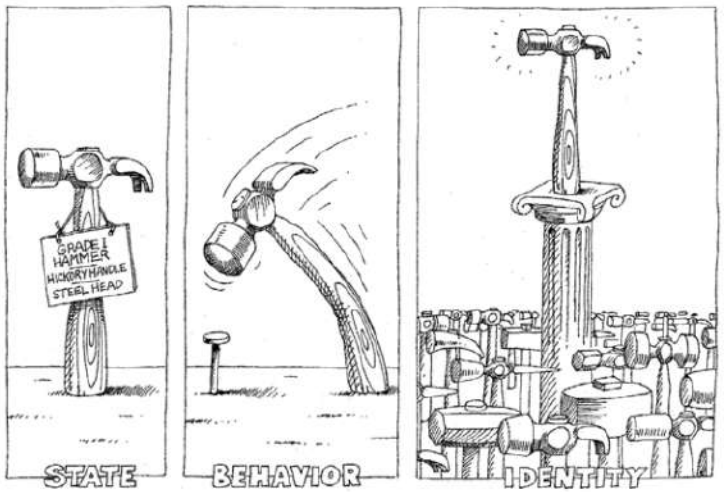
- write setters only for modifiable fields (logically)
- write getters only for fields to be read
- setters can have checks for valid values


java.lang.Object
java.lang.Boolean
<<final>>
java.lang.Number
<<Abstract>>
java.lang.Character
<<final>>
java.lang.Byte
<<final>>
java.lang.Short
<<final>>
java.lang.Integer
<<final>>
java.lang.Float
<<final>>
java.lang.Double
<<final>>
java.lang.Long
<<final>>

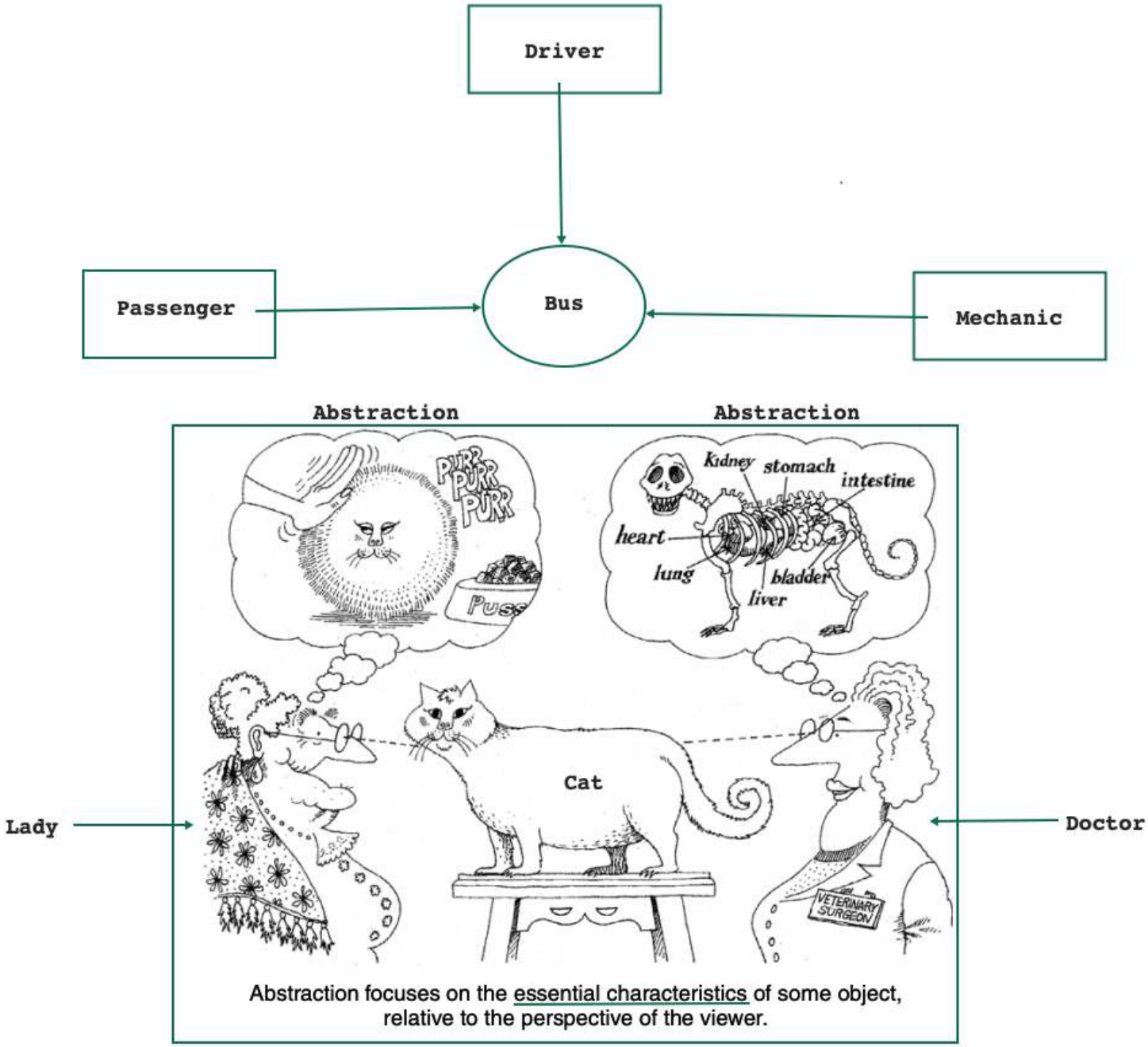
```
Employee emp = new Employee( );
```

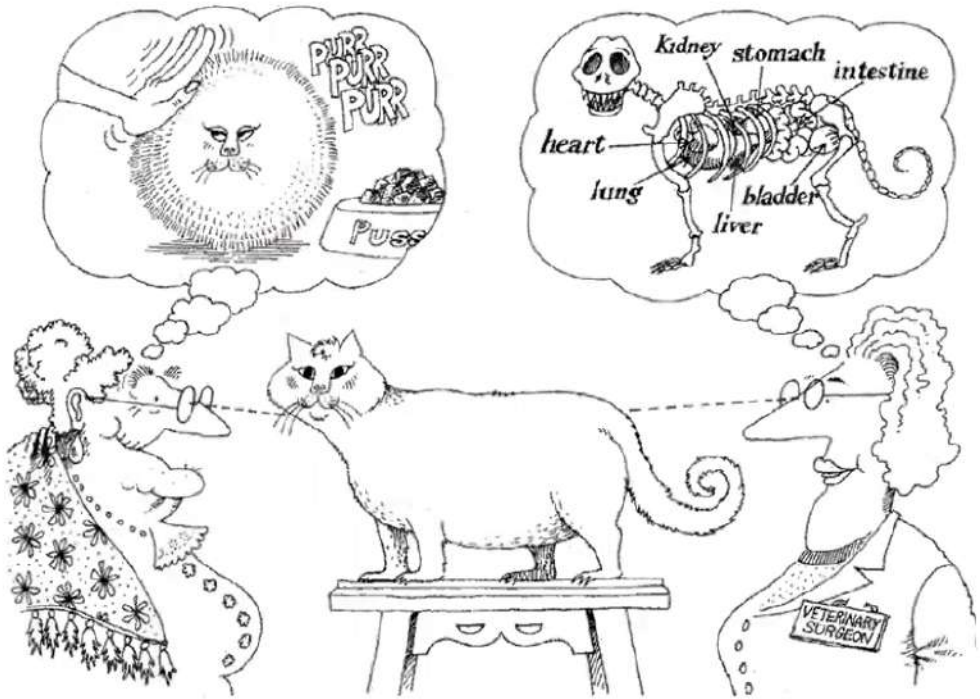



Employee emp = new Employee();
[Stack]
[Heap]
emp
Employee
[Employee class reference]
name
String
empid
int
salary
float

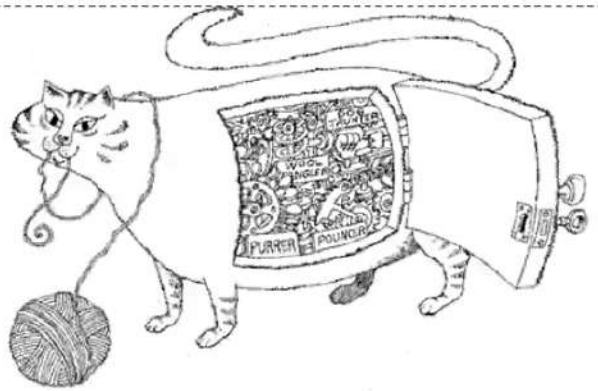


An object has state, exhibits some well-defined behavior, and has a unique identity.


Driver
Passenger
Bus
Mechanic
Abstraction
Abstraction
PURR PURR PURR
Puss
Kidney
stomach
intestine
heart
lung
bladder
liver
Cat
Lady
Doctor
VETERINARY SURGEON
Abstraction focuses on the essential characteristics of some object,
relative to the perspective of the viewer.



Abstraction focuses on the essential characteristics of some object, relative to the perspective of the viewer.



Encapsulation hides the details of the implementation of an object.

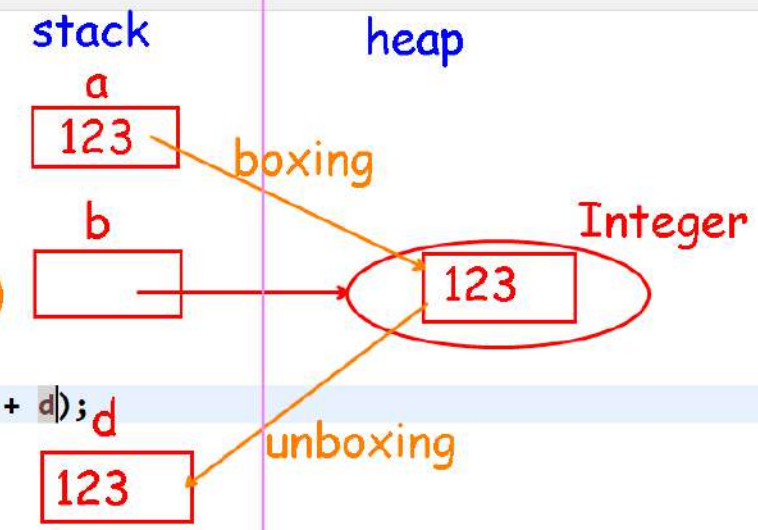
Abstraction and encapsulation are complementary concepts: Abstraction focuses on the observable behavior of an object, whereas encapsulation focuses on the implementation that gives rise to this behavior. Encapsulation is most often achieved through information hiding (not just data hiding), which is the process of hiding all the secrets of an object that do not contribute to its essential characteristics; typically, the structure of an object is hidden, as well as the implementation of its methods. "No part of a complex system should depend on the internal details of any other part" [50]. Whereas abstraction "helps people to think about what they are doing," encapsulation "allows program changes to be reliably made with limited effort" [51].

day02 - demo04/src/Program02.java - Spring Tool Suite 4

File Edit Source Refactor Navigate Search Project Run Window Help

Program02.java

```java
public class Program02 {
    public static void main(String[] args) {
        int a = 123;
        // convert primitive int to wrapper Integer (boxing)
        Integer b = new Integer(a);

        // convert wrapper Integer to primitive int (unboxing)
        int d = b.intValue();

        System.out.println("a = " + a + ", b = " + b + ", d = " + d);
    }
}
```



Problems Javadoc Declaration Console

<terminated> Program02 (2) [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20220515-1416\jre\bin\javaw.exe (Jan 30, 2024, 12:53:32 PM – 12:53:32 PM) [pid: 3176]

```
a = 123, b = 123, d = 123
```

Writable Smart Insert 11 : 68 : 306

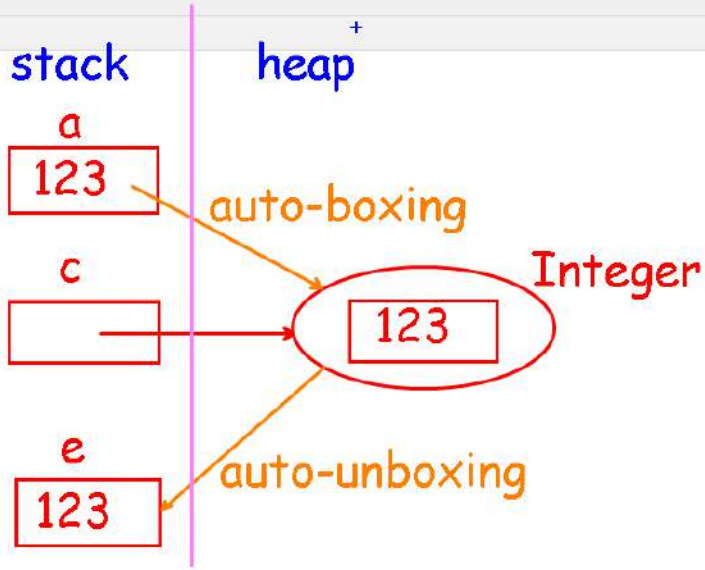
Search 12:53 PM

day02 - demo04/src/Program02.java - Spring Tool Suite 4

File Edit Source Refactor Navigate Search Project Run Window Help

Program02.java

```java

public class Program02 {
    public static void main(String[] args) {
        int a = 123;
        // convert primitive int to wrapper Integer -- boxing
        Integer b = new Integer(a);
        // convert primitive int to wrapper Integer -- auto-boxing
        Integer c = a;

        // convert wrapper Integer to primitive int
        int d = b.intValue();
        // convert wrapper Integer to primitive int -- auto-unboxing
        int e = c;

        System.out.println("a = " + a + ", b = " + b + ", c = " + c + ", d = " + d + ", e = " + e);
    }
}

```



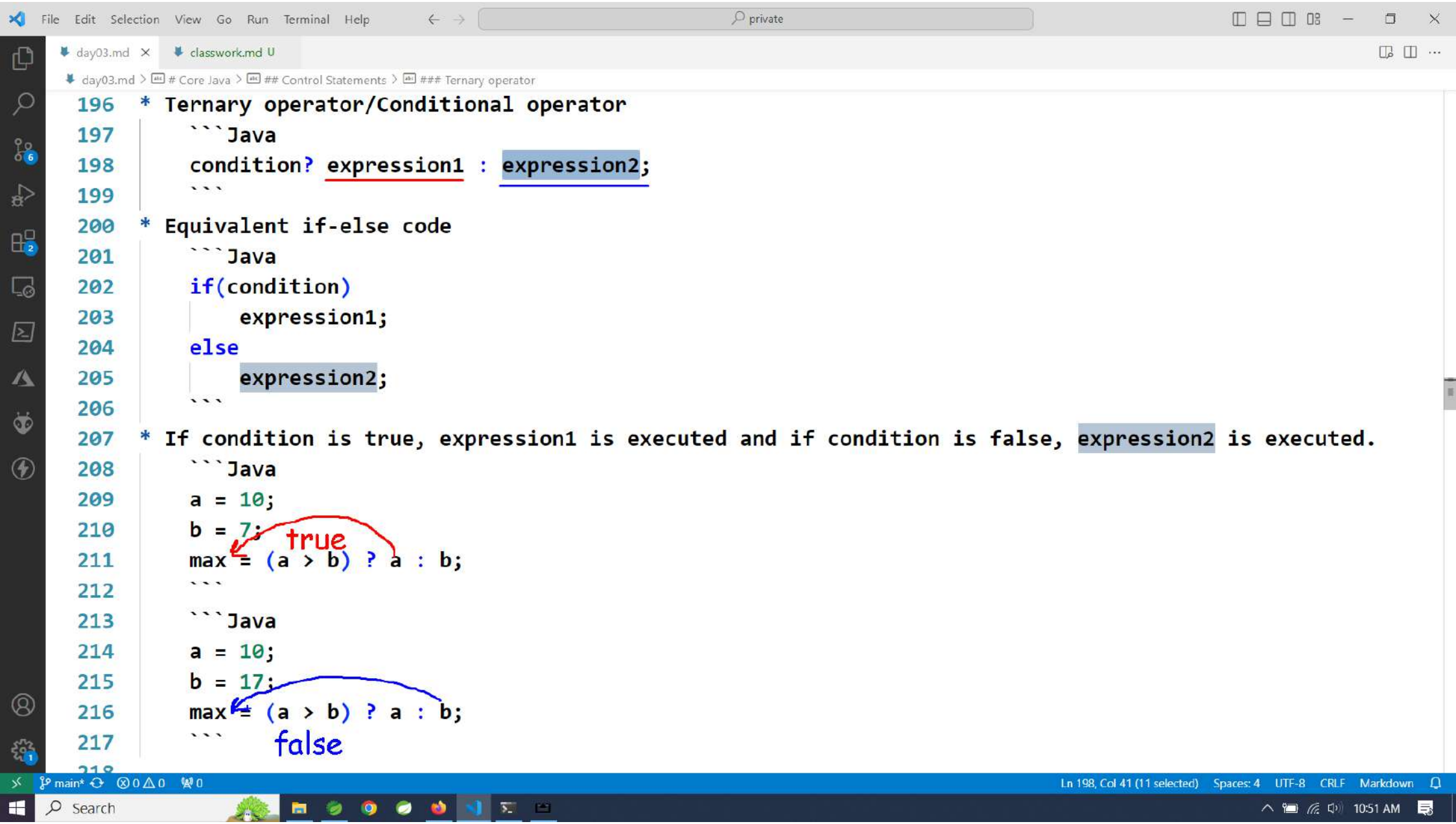
Problems Javadoc Declaration Console

<terminated> Program02 (2) [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20220515-1416\jre\bin\javaw.exe (Jan 30, 2024, 12:58:53 PM – 12:58:53 PM) [pid: 4172]

```
a = 123, b = 123, c = 123, d = 123, e = 123
```

Writable Smart Insert 15 : 100 : 506

Search 12:59 PM


File Edit Selection View Go Run Terminal Help
private
day03.md
classwork.md U
day03.md > # Core Java > ## Control Statements > ### Ternary operator
196 * Ternary operator/Conditional operator
197 ```Java
198 condition? expression1 : expression2;
199 ```
200 * Equivalent if-else code
201 ```Java
202 if(condition)
203 expression1;
204 else
205 expression2;
206 ```
207 * If condition is true, expression1 is executed and if condition is false, expression2 is executed.
208 ```Java
209 a = 10;
210 b = 7;
211 max = (a > b) ? a : b;
212 ```
213 ```Java
214 a = 10;
215 b = 17;
216 max = (a > b) ? a : b;
217 ```
true
false
main*
Ln 198, Col 41 (11 selected)
Spaces: 4
UTF-8
CRLF
Markdown
Search
10:51 AM

Class and Object

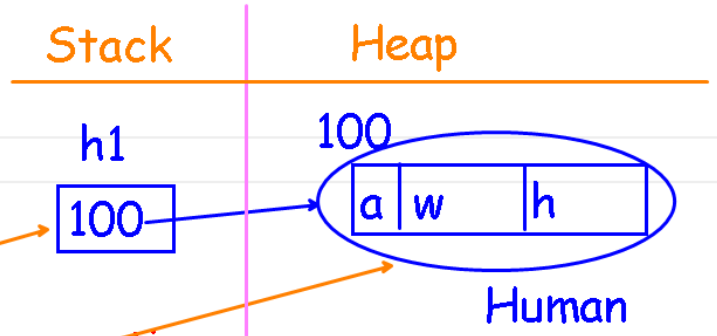
* Class is collection of logically related data members ("fields"/attributes/properties) and the member functions ("methods"/operations/messages) to operate on that data.
* A class is user defined data type. It is used to create one or more instances called as "Objects".
* Class is blueprint/prototype/template of the object; while Object is an instance of the class.
* Class is logical entity and Object represent physical real-world entity.
    * e.g. Human is a class and You are one of the object of the class.

```Java
class Human {
    int age;
    double weight;
    double height;
    // ...
    void walk() { ... }
    void talk() { ... }
    void think() { ... }
    // ...
}
```

Human h1 = new Human(); -- object creation

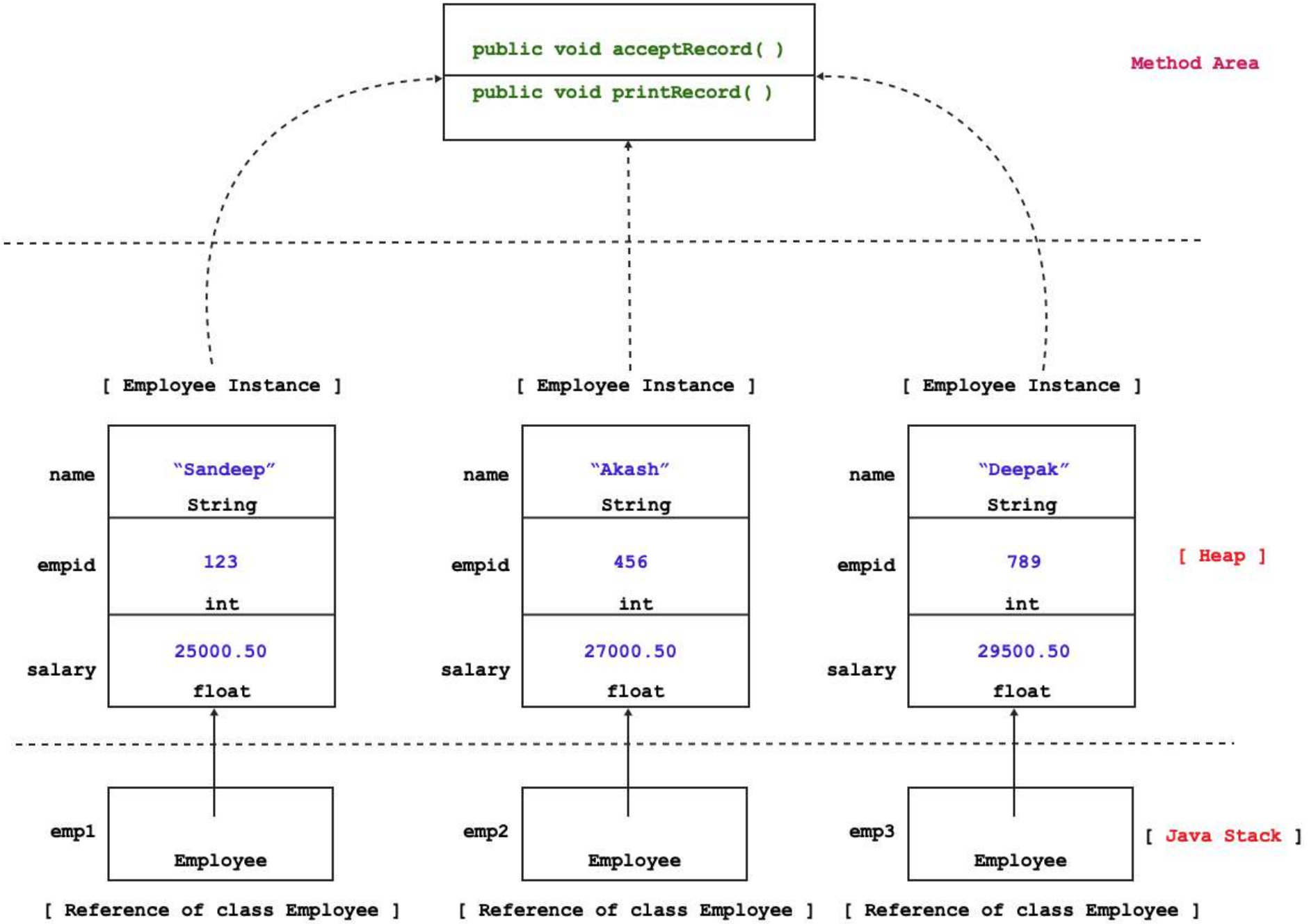
data members / fields / attributes

member functions / methods / operations



* Since class is non-primitive/reference type in Java, its objects are always created on heap (using new operator). Object creation is also referred as "Instantiation" of the class.

```Java
```


public void acceptRecord( )
public void printRecord( )
Method Area
[Employee Instance]
[Employee Instance]
[Employee Instance]
name
"Sandeep"
String
empid
123
int
salary
25000.50
float
name
"Akash"
String
empid
456
int
salary
27000.50
float
name
"Deepak"
String
empid
789
int
salary
29500.50
float
[Heap]
emp1
Employee
emp2
Employee
emp3
Employee
[Java Stack]
[Reference of class Employee]
[Reference of class Employee]
[Reference of class Employee]