Concepts of Programming in Java

- Akshita Chanchlani

Class

- Consider following examples:
  1. day, month, year - related to – Date
  2. hour, minute, second - related to – Time
  3. red, green, blue - related to Color
  4. real, imag - related to – Complex
  5. xPosition, yPosition - related to Point
  6. number, type, balance - related to Account
  7. name, id, salary - related to Employee

- If we want to group related data elements together then we should use/define class in Java.

```
class Date{
    int day;        //Field
    int month;      //Field
    int year;       //Field
}
```

```
class Employee{
    String name;    //Field
    int id;         //Field
    float salary;   //Field
}
```

Class

- class is a non primitive/reference type in Java.
- If we want create object/instance of a class then it is mandatory to use new operator.
- If we create instance using new operator then it gets space on heap section.
- Only fields of the get space once per instance according to order of their declaration inside class.

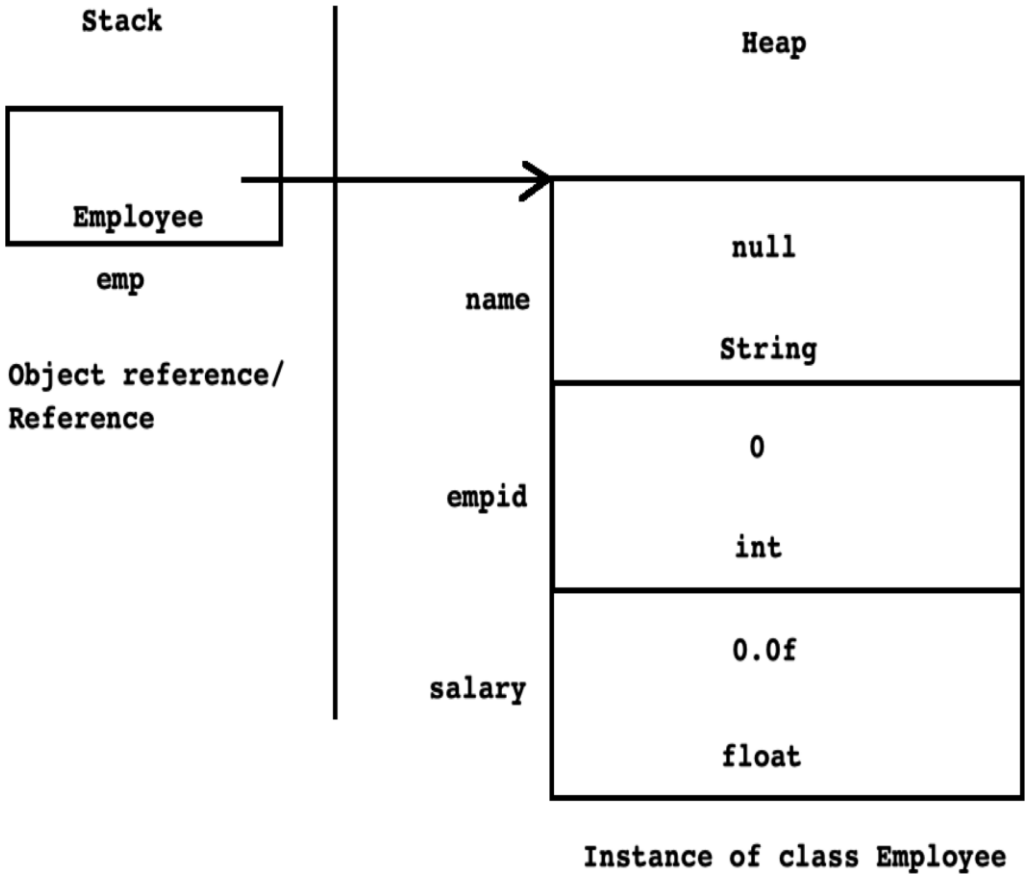
Class

- Field
  - Ø A variable declared inside class / class scope is called a field.
  - Ø Field is also called as attribute or property.
- Method
  - Ø A function implemented inside class/class scope is called as method.
  - Ø Method is also called as operation, behavior or message.
- Class
  - Ø Class is a collection of fields and methods.
  - Ø Class can contain
    1. Nested Type
    2. Field
    3. Constructor
    4. Method
  - **Instance**
    - Ø In Java, Object is also called as instance.

Instantiation

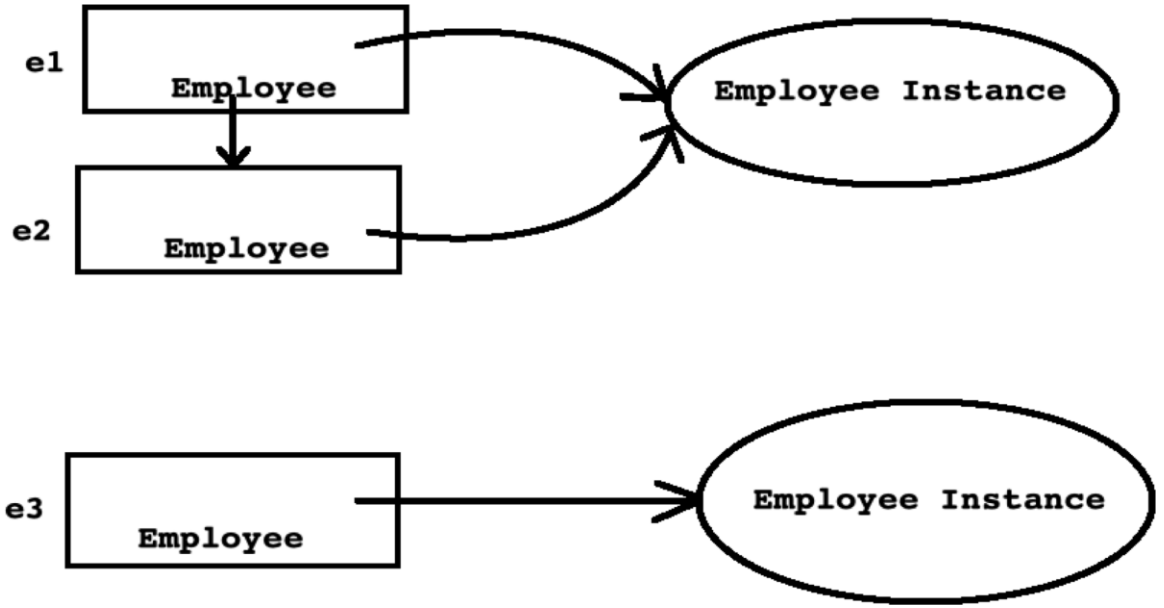
- Process of creating instance/object from a class is called as instantiation.
- In C programming language
  - Ø Syntax : struct StructureName identifier_name; struct
  - Ø Employee emp;
- In C++ programming language
  - Ø Syntax : [class] ClassName identifier_name;
  - Ø Employee emp;
- In Java programming language
  - Ø Syntax : ClassName identifier_name = new ClassName( );
  - Ø Employee emp = new Employee();
- **Every instance on heap section is anonymous.**

Instantiation



For eg :

```
1. Employee e1 = new Employee();
2. Employee e2 = e1;
3. Employee e3 = new Employee();
```



this reference

- If we call non static method on instance(actually object reference) then compiler implicitly pass, reference of current/calling instance as a argument to the method implicitly. To store reference of current/calling instance, compiler implicitly declare one reference as a parameter inside method. It is called this reference.

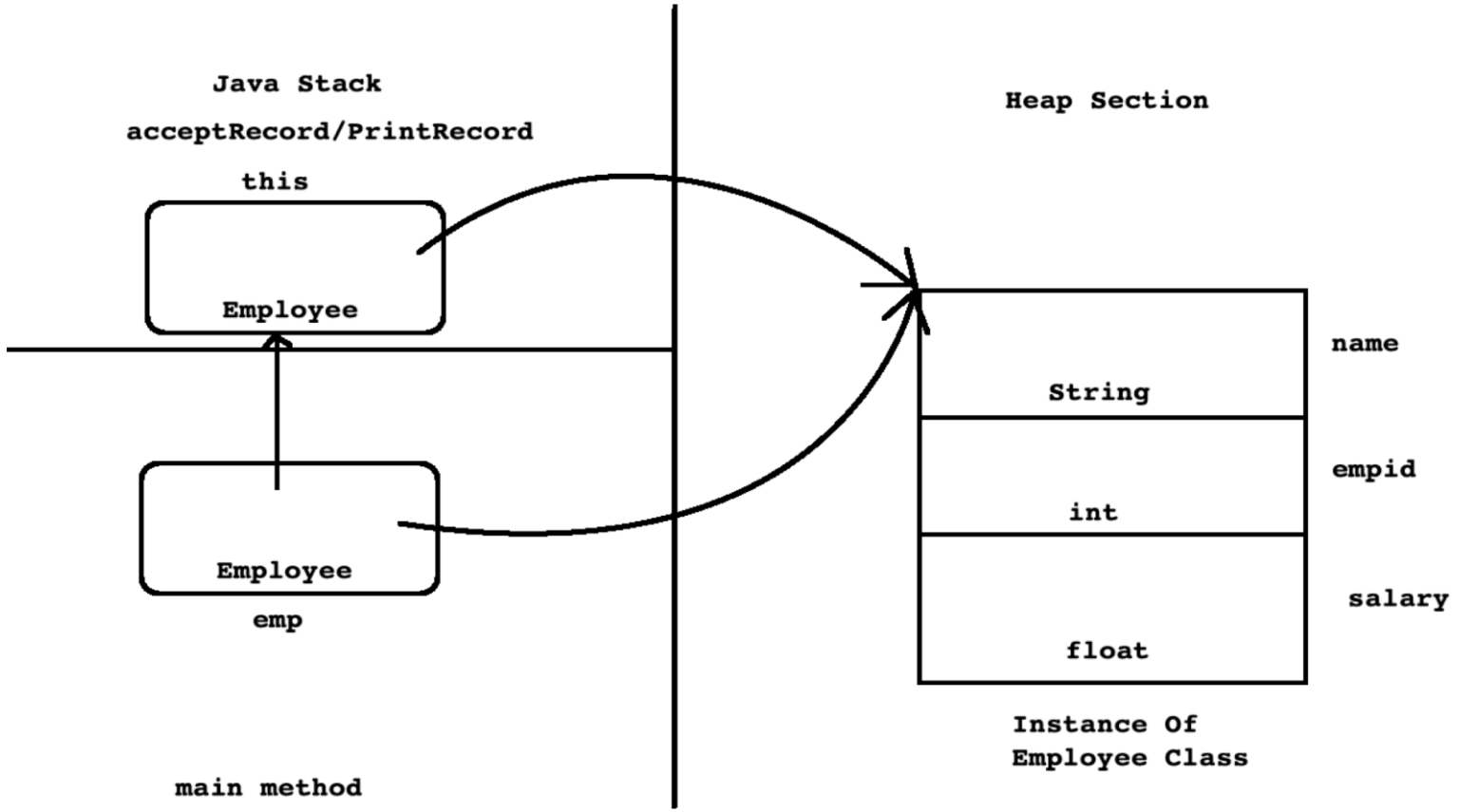
- **this is a keyword** in Java which is designed to store reference of current/calling instance.

- **Using this reference, non static fields and non static methods are communicating with each other. Hence this reference is considered as a link/connection between them.**

- Definition
  - Ø **"this" is implicit reference variable that is available in every non static method of class which is used to store reference of current/calling instance.**

- Inside method, to access members of same class, use this keyword is optional

this reference



this reference

- If name of local variable/parameter and name of field is same then preference is always given to the local variable.

```
class Employee{
    private String name;
    private int empid;
    private float salary;
    public void initEmployee(String name, int empid, float salary ){
        this.name = name;
        this.empid = empid;
        this.salary = salary;
    }
}
```

Constructor

- If we want to initialize instance then we should define constructor inside class.
- Constructor look like method but it is not considered as method.
- It is special because:
  - Its name is same as class name.
  - It doesn't have any return type.
  - It is designed to be called implicitly
  - It is called once per instance.
- We can not call constructor on instance explicitly

  Employee emp = new Employee();
  emp.Employee( );//Not Ok

- **Types of constructor:**
  1. Parameterless constructor
  2. Parameterized constructor
  3. Default constructor .

Parameterless Constructor

- If we define constructor without parameter then it is called as parameterless constructor.
- It is also called as zero argument / user defined default constructor.
- If we create instance without passing argument then parameterless constructor gets called.

```
public Employee( ){
    //TODO
}
```

```
Employee emp = new Employee( ); //Here on instance parameterless ctor will call.
```

Parameterized Constructor

- If we define constructor with parameter then it is called as parameterized constructor.
- If we create instance by passing argument then parameterized constructor gets called.

```
public Employee( String name, int empid, float salary ){
    //TODO
}
```

```
Employee emp = new Employee( "ABC",123, 8000 ); //Here on instance parameterized ctor will call.
```

Default Constructor

- If we do not define any constructor inside class then compiler generate one constructor for the class by default. It is called default constructor.

- Compiler generated default constructor is parameterless.
- Compiler never generate default parameterized constructor. In other words, if we want to create instance by passing arguments then we must define parameterized constructor inside class.

Constructor Chaining

- We can call constructor from another constructor. It is called constructor chaining.
- For constructor chaining, we should use this statement.
- this statement must be first statement inside constructor body.
- Using constructor chaining, we can reduce developers effort.

```
class Employee{
    //TODO : Field declaration
    public Employee( ){
        this( "None", 0, 8500 );    //Constructor Chaining
    }
    public Employee( String name, int empid, float salary ){
        this.name = name;
        this.empid = empid;
        this.salary = salary;
    }
}
```

Literals

- Consider following literals in Java:
  1. true : boolean
  2. 'A' : char ch;
  3. “Akshita" : String str;
  4. 123 : int num1;
  5. 72.93f : float num2
  6. 3.142 : double num3
  7. null : Used to initialize reference variable.

- null is a literal which is designed to initialize reference variable
  - int num = null ; //invalid
  - Integer num=null; // VALID
  - String str=null; // VALID
  - Employee emp=null; // VALID

null

If reference variable contains null value then it is called null reference variable / null object.

```
class Program{
    public static void main(String[] args) {
      Employee emp; //Object reference / reference
      emp.printRecord(); //error: variable emp might not have been initialized
    }
}
```

```
public static void main(String[] args) {
    Employee emp = null; //null reference varibale / null object
    emp.printRecord();    //NullPointerException
}
```

Value type VS Reference Type

Sr. No.	Value Type	Reference Type
1	primitive type is also calle as value type.	Non primitive type is also called as reference type.
2	boolean, byte, char, short, int, float, double, long are primitive/value type.	Interface, class, type variable and array are non primitive/reference type.
3	Variable of value type contains value.	Variable of reference type contains reference.
4	Variable of value type by default contains 0 value.	Variable of reference type by default contain null reference.
5	We can not create variable of value type using new operator.	It is madatory to use new operator to create instance of reference type.
6	variable of value type get space on Java stack.	Intstance of reference type get space on heap section.
7	We can not store null value inside variable of value type.	We can store null value inside variable reference type.
8	In case of copy, value gets copied.	In case of copy, reference gets copied.

Coding Convention

- **Pascal Case Coding/Naming Convention:**
  - Example
    1. System
    2. StringBuilder
    3. NullPointerException
    4. IndexOutOfBoundsException

  - In this case, including first word, first character of each word must in upper case.

  - We should use this convention for:
    1. Type Name(Interface, class, Enum, Annotation)
    2. File Name

Coding Convention

- **Camel Case Coding/Naming Convention:**
  - Example
    1. main
    2. parseInt
    3. showInputDialog
    4. addNumberOfDays
  - In this case, excluding first word, first character of each word must in upper case.
  - We should use this convention for:
    1. Method Parameter and Local variable
    2. Field
    3. Method
    4. Reference

Coding Convention

- **Naming Convention for package:**
  - We can specify name of the package in uppercase as well as lower-case. But generally it is mentioned in lower case.
  - Example
    1. java.lang
    2. java.lang.reflect
    3. java.util
    4. java.io
    5. java.net
    6. java.sql

Coding Convention

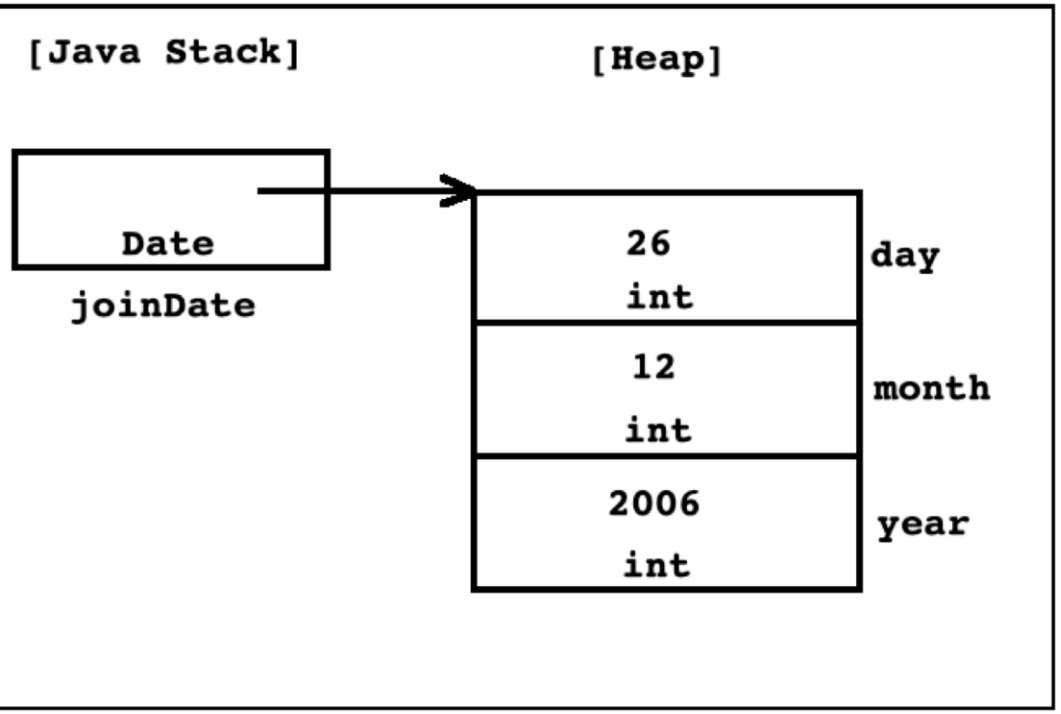
- **Naming Convention for constant variable and enum constant:**
  - Example

```
1. public static final int SIZE;
2. enum Color{ RED, GREEN, BLUE }
3. Name of the final variable and name of the enum constant should be in upper case.
```

Reference

- Local reference variable get space on Java Stack.

```
class Date{
    private int day;
    private int month;
    private int year;
}
class Program{
    public static void main(String[] args) {
        Date joinDate = new Date( 26,12,2006);
        //joinDate : Method Local Variable
    }
}
```



- In above code joinDate is method local reference variable hence it gets space on Java Stack.

Reference

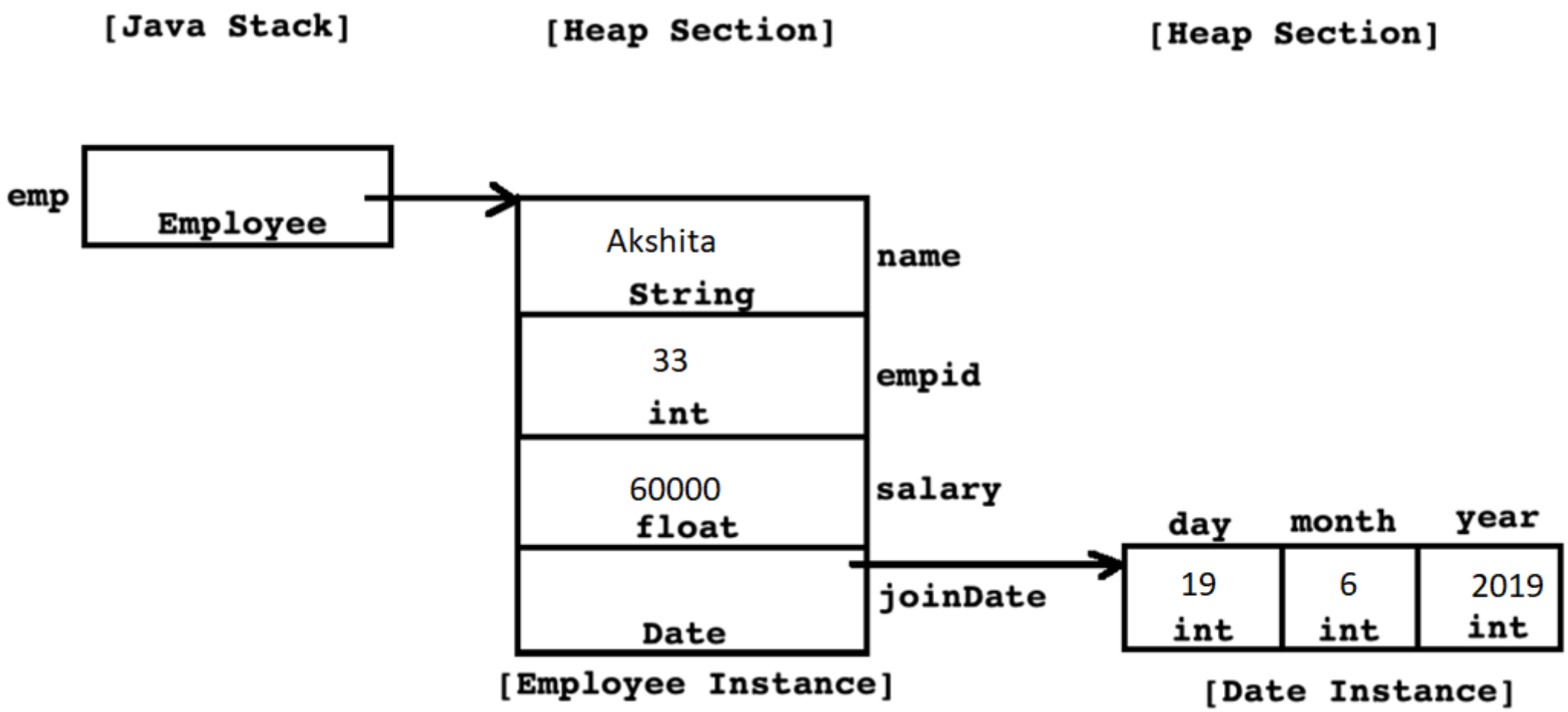
- Class scope reference variable get space on heap.

```
class Employee{
    private String name;
    private int empid;
    private float salary
    private Date joinDate;  //joinDate : Field
    public Employee( String name, int empid, float salary, Date joinDate ){
        this.name = name;
        this.empid = empid;
        this.salary = salary;
        this.joinDate = joinDate;
    }
}
```

```
Class Program
{
 public static void main(String[] args)
{
    Employee emp = new Employee (“Akshita”,33,60000,new Date(19,06,2019));
}
}
```

- In above code, emp is method local reference variable hence it gets space on Java Stack. But joinDate is field of Employee class hence it will get space inside instance on Heap.

Reference


[Java Stack]
[Heap Section]
[Heap Section]
emp
Employee
Akshita
String
name
33
int
empid
60000
float
salary
Date
joinDate
day
month
year
19
int
6
int
2019
int
[Employee Instance]
[Date Instance]

Thank you.

akshita.Chanchlani@sunbeaminfo.com