Java Standard Edition 8

Sandeep Kulange
sandeepkulange@sunbeaminfo.com

Steps to write code in Java.

- Understand problem statement and analyze it.
- Decide class and fields for it. A variable declared inside class scope is called field.
- Create instance of a class(Instantiate Java class).
    - ➢ Process of creating instance/object from a class is called as instantiation.
- Following members of a class do not get space inside instance:
    1. Nested Type(Interface/Class/Enum)
    2. Static Field
    3. Constructor
    4. Method(static/non static)
    5. Method Parameter and Local variable.
- If we create instance of a class then only non static fields get space inside it.
- If we want to process state/value of instance/object then we should define and invoke method on instance. This process of calling method on instance is called message passing.

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.

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

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

Instantiation

- Process of creating instance/object from a class is called as instantiation.

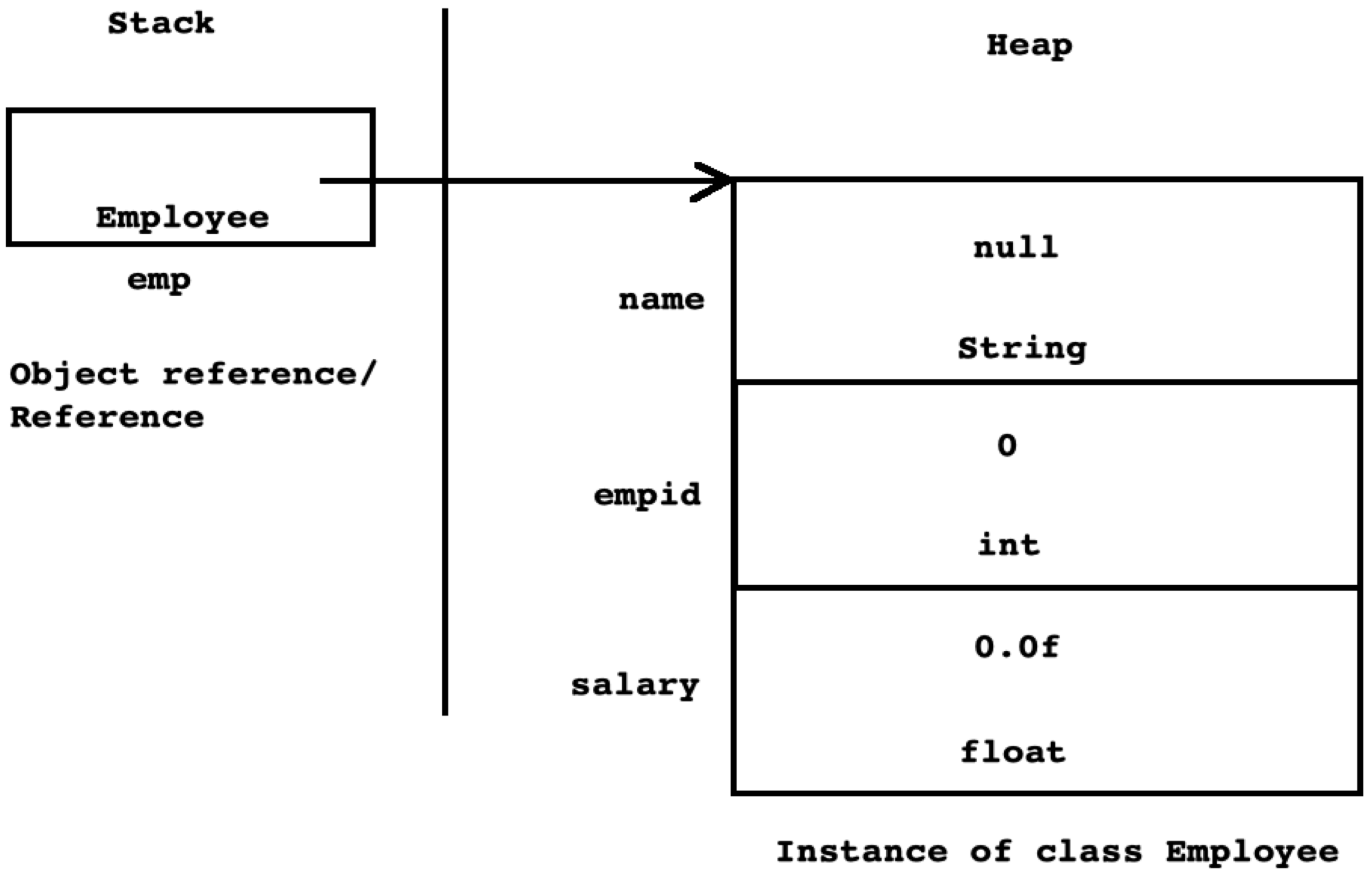
- In C programming language
  - ➢ Syntax : struct StructureName identifier_name;
  - ➢ struct Employee emp;

- In C++ programming language
  - ➢ Syntax : [class] ClassName identifier_name;
  - ➢ Employee emp;

- In Java programming language
  - ➢ Syntax : ClassName identifier_name = new ClassName( );
  - ➢ Employee emp = new Employee();

- **Every instance on heap section is anonymous.**

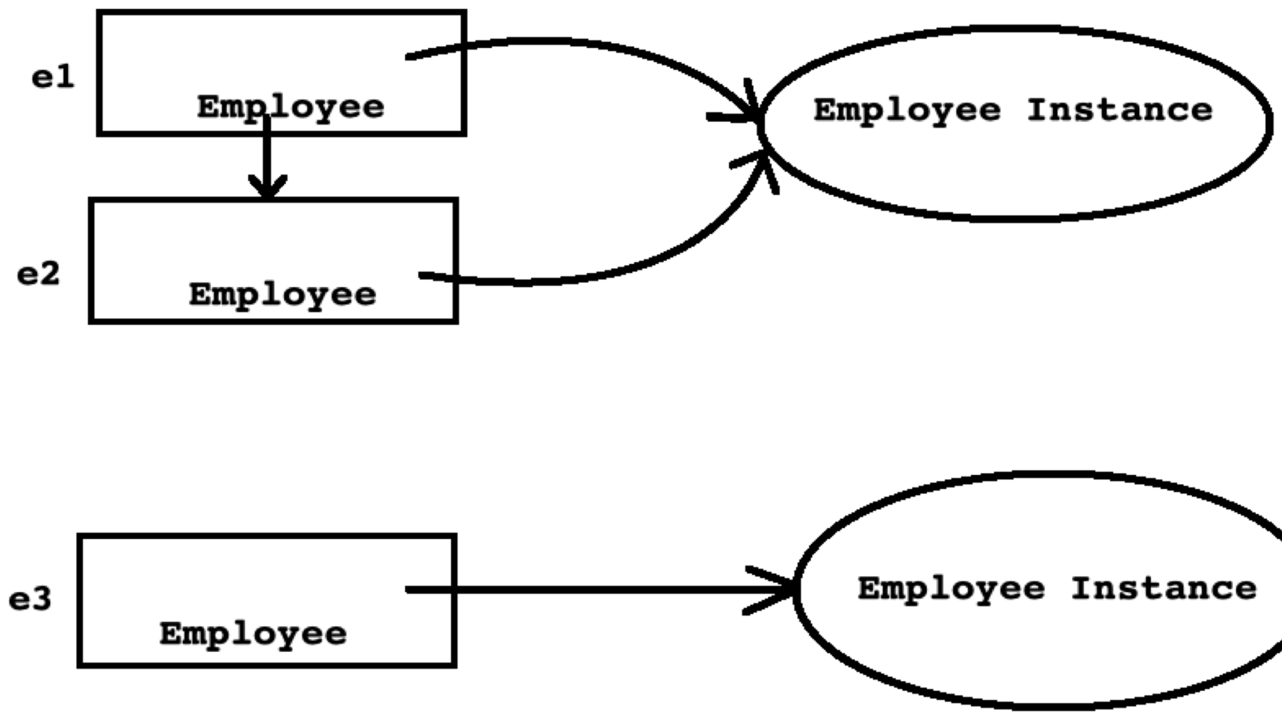
Instantiation



Instantiation

- Consider following code:

```
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.

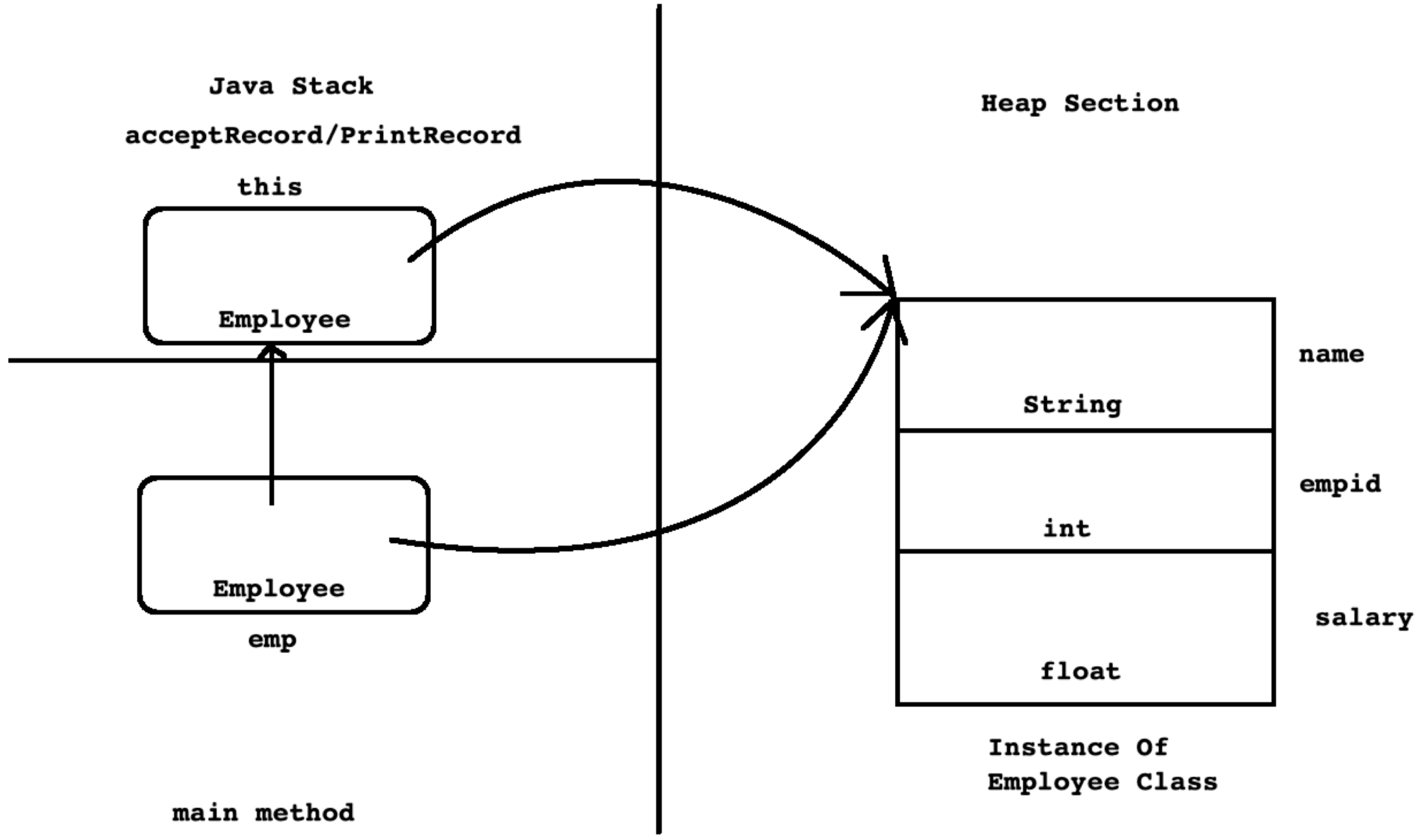
- **this is a keyword** in Java which is designed to store reference of current/calling instance.

- **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
  - **"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.**

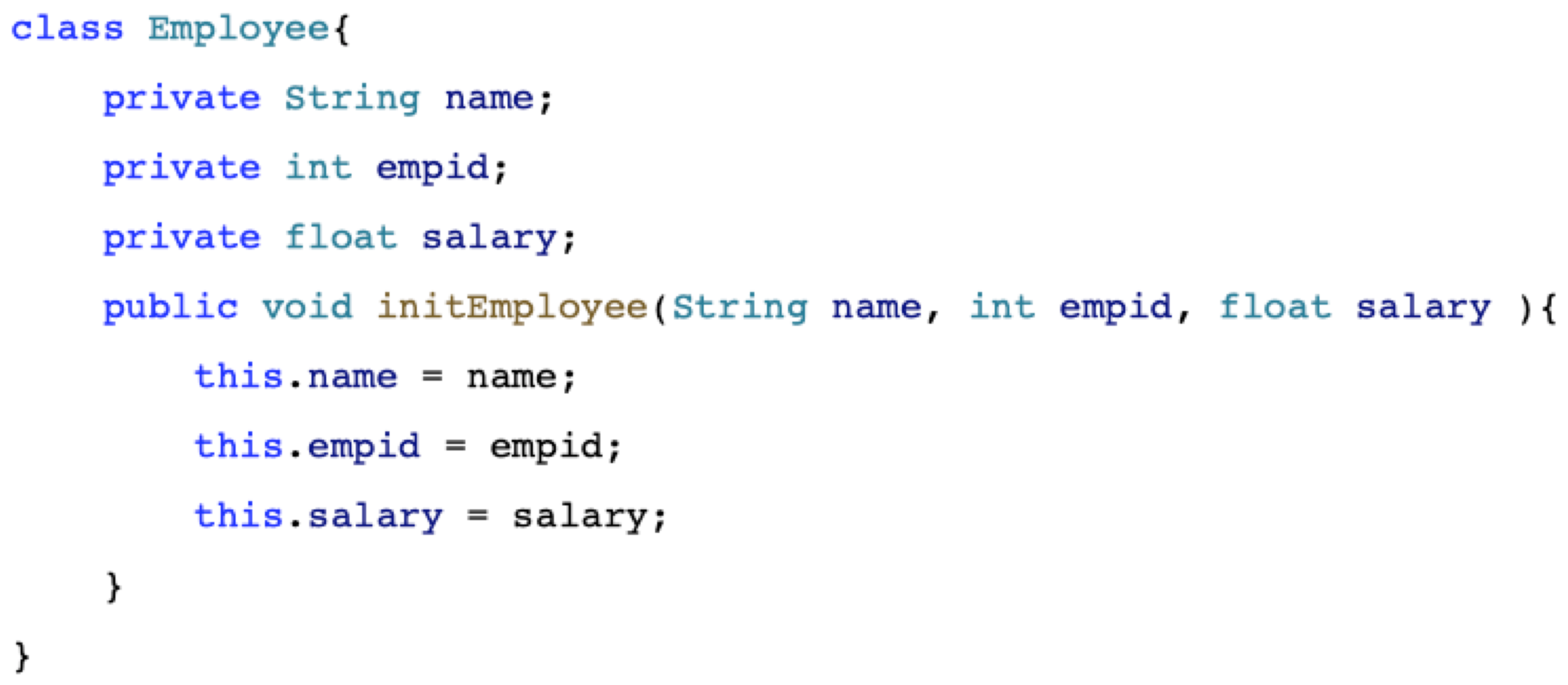
- Inside method, to access members of same class, use this keyword is optional

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:
  1. Its name is same as class name.
  2. It doesn't have any return type.
  3. It is designed to call implicitly.
  4. It gets called once per instance.

- We can not call constructor on instance explicitly

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

Constructor

- We can use any access modifier on constructor.
- If constructor is public then we can create instance of a class inside method of same class as well as different class.
- If constructor is private then we can create instance of a class inside method of same class only.
- 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.

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

- If we create instance without passing argument then parameterless constructor gets called.

```
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.

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

- If we create instance by passing argument then parameterized constructor gets called.

```
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.

```
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;
    }
}
```

- Using constructor chaining, we can reduce developers effort.

Literals

- Consider following literals in Java:
  1. true : boolean
  2. 'A' : char ch;
  3. "Sandeep" : 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 number = null;   //Not OK
Integer n1 = null;   //OK
String str = null;   //OK
Employee emp = null;     //OK
```

null Literal

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

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

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

Thank you