Object Oriented Programming With Java

Day3 Agenda

- Class
- Steps for write code for class
- Reference and Instance
- this reference
- Constructor
- Constructor Chaining
- What is null
- Value Type Versus Reference Type.

Major and minor pillars of oops

- 4 Major pillars of oops:
  1. Abstraction
  2. Encapsulation
  3. Modularity
  4. Hierarchy

- Must be required in object oriented programming language.

- 3 minor pillars of oops
  1. Typing/Polymorphism
  2. Concurrency
  3. Persistence

- Useful but not required in object oriented programming language.

Class

- If we want to group functionally equivalent / related elements together then we should use class
- Consider Examples
  - Date
    - Related data elements are day, month and year
  - Address
    - Related data elements are cityname, statename and pincode
  - Color
    - Related data elements are red, green and blue
  - Student
    - Related data elements are name, rollnumber and marks
  - Department
    - Related data elements are id, name and location.
  - Employee
    - Related data elements are name, empid and salary

Class

- class is a keyword in Java which can contain:
    1. Nested types(interface, class, enum)
    2. Field
    3. Constructor
    4. Method

- We can define class inside class it is called nested class.

- Variable declared inside class is called as field.

- Function implemented inside class is called method.

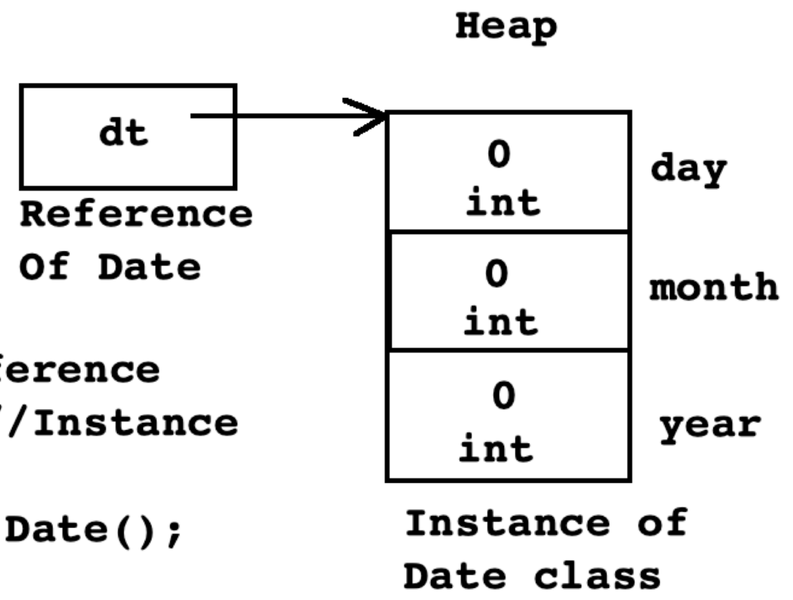
- Constructor is used to initialize instance.

Consider steps while writing a class

1. Understand assignment/problem statement and decide classes and its fields.
2. Define class and declare fields inside it.
3. Create Instance of class.
   - To create instance of a class it is mandatory to use new operator.
   - Employee emp = new Employee( );
   - If we create instance of a class then fields get space inside it.
4. To process(accept/print/set/get) state/value of instance define and invoke method on it.
   - Process of calling method on instance is called message passing.
5. Use this reference inside method to process state of instance.

Reference and Instance

+ non static field gets space once per instance according to order of their declaration inside class.



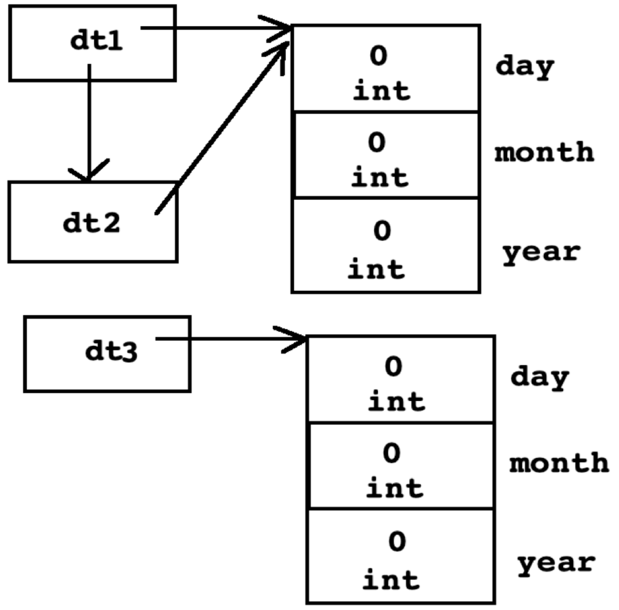
```
Date dt; //reference
new Date();   //Instance

Date dt = new Date();
```

+ If we want to process state of instance(call method on instance) then we should create reference of a class.

Reference and Instance

```
Date dt1 = new Date( );
Date dt2 = dt1;
Date dt3 = new Date( );
```

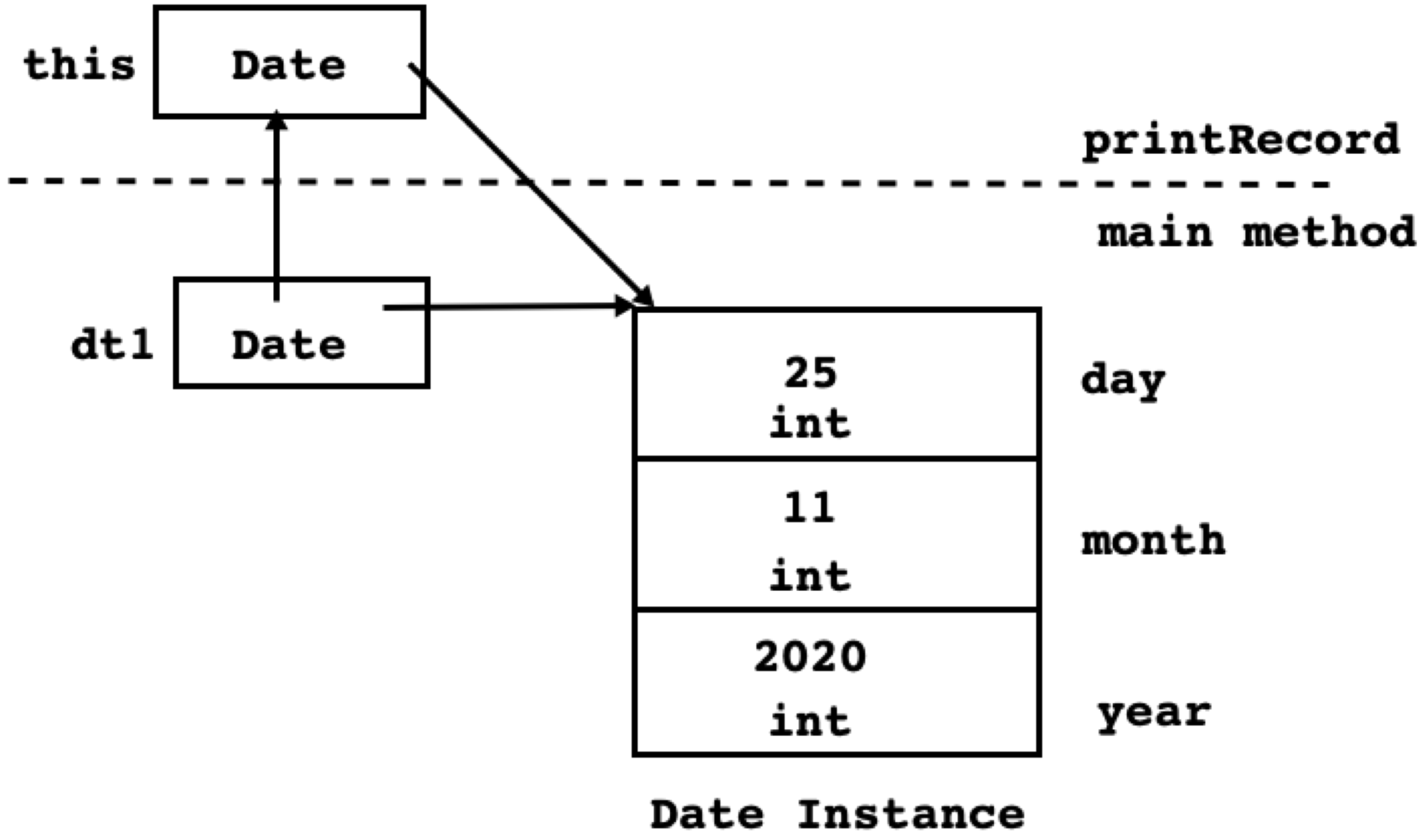


this reference

```
class Date{
        private int day, month, year;
        public void printRecord( /*Date this*/ ){
                //TODO : use this to access state of Date instance.
        }
}
class Program{
        public static void main( String[] args ){
                Date dt1 = new Date( );
                dt1.printRecord( );        // dt1.printRecord( dt1 );

                Date dt2 = new Date( );
                dt2.printRecord( );        // dt2.printRecord( dt2 );
        }
}
```

this reference



this reference

- To process value or state of instance, we should call method on instance.
- If we call non static method on instance then compiler implicitly pass reference of current instance as a argument. And to store value of the argument compiler implicitly declare one parameter inside method. It is called this reference.
- this is a keyword in Java.
- Using this reference, non static field and non static method can communicate with each other. Hence it is also called as link /connection between non static field and non static method.
- We can not declare this reference explicitly. It is implicit parameter available in every method of a class. It is considered as first parameter of a method.

- In simple words, It is a implicit reference variable which is available in every non static method of a class which is used to store reference of current or calling instance.

- If name of local variable and name of field is same then we should use this before field.

Constructor

- If we want to initialize instance then we should use constructor.
- Constructors are 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.

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

Parameterless Constructor

- A constructor which do not take any parameter is called parameterless constructor.
- Consider Example:

```
class Date{
    private int day, month, year;
    public Date( ){      //Parameterless constructor
            this.day = 25;
            this.month = 11;
            this.year = 2020;
    }
}
```

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

```
Date dt = new Date( ); //Here parameterless constructor will call
```

Parameterized Constructor

- A constructor which take parameter(s) is called parameterized constructor.
- Consider Example:

```
class Date{
    private int day, month, year;
    public Date( int day, int month, int year ){   //Parameterized constructor
            this.day = day;
            this.month = month;
            this.year = year;
    }
}
```

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

```
Date dt = new Date( 25,11,2020); //Here parameterized constructor will call
```

Default Constructor

- If we do not define constructor(no constructor) inside a class then compiler generates once constructor for the class by default it is called default constructor.
- The default constructor is zero argument constructor.

- Consider Example:

```
class Date{
   private int day, month, year;
   public void print( ){
   }
}
```

- Instantiation

```
   Date dt1 = new Date( ); //Here Default constructor will call
   Date dt2 = new Date( 25, 11, 2020 ); //Compiler Error
```

Constructor Chaining

- If class contains set of constructors then constructor can reuse logic of another constructor.
- If we want to reuse implementation of existing constructor inside another constructor then we should call constructor explicitly. It is called constructor chaining.
- For constructor chaining we should use this statement.
- Consider Example:

```
class Date{
    private int day, month, year;
    public Date( ){
        this(25,11,2020);    //Constructor Chaining
    }
    public Date( int day, int month, int year ){
        this.day = day;
        this.month = month;
        this.year = year;
    }
}
```

- “this” statement must be first statement inside constructor body.

Literals

- Constant is also called as literal.
- Consider examples:

1.	true	:	Use boolean to store value
2.	‘A’	:	Use char to store value
3.	123	:	Use byte/short/int/long to store value
4.	3.14f	:	Use float to store value
5.	3.142	:	Use double to store value
6.	“SunBeam”	:	Use String to store value
7.	null	:	Use reference to store value.

- NULL is macro in C which is used to initialize pointer.
- Example : int *ptr = NULL;
- null is literal in Java which is used to initialize reference variable.
- Example : Employee emp = null;

Value Type versus Reference Type

Value Type

1. Primitive type is also called as value type.
2. There are 8 values types in Java.
3. Variable of value type get space on Java Stack.
4. Variable of value type contain value.
5. In case of copy, value gets copied.
6. Variable of value type do not contain null.
7. We can not use new operator to create instance of value type.

Reference Type

1. Non primitive type is also called reference type.
2. There are 4 reference types in Java.
3. Instance of reference type get space on Heap.
4. Variable of reference contain reference.
5. In case of reference, reference gets copied.
6. Variable of reference type can contain null.
7. It is mandatory to use new operator to create instance of reference type.

Thank you