Writing Class in Java

Session 3

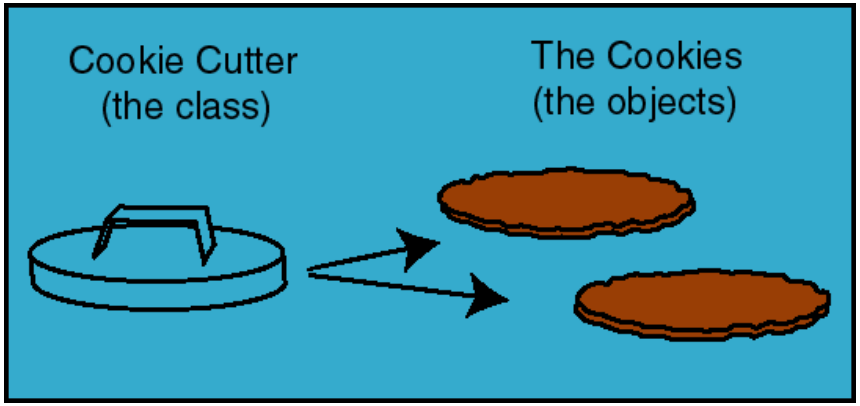
Contents

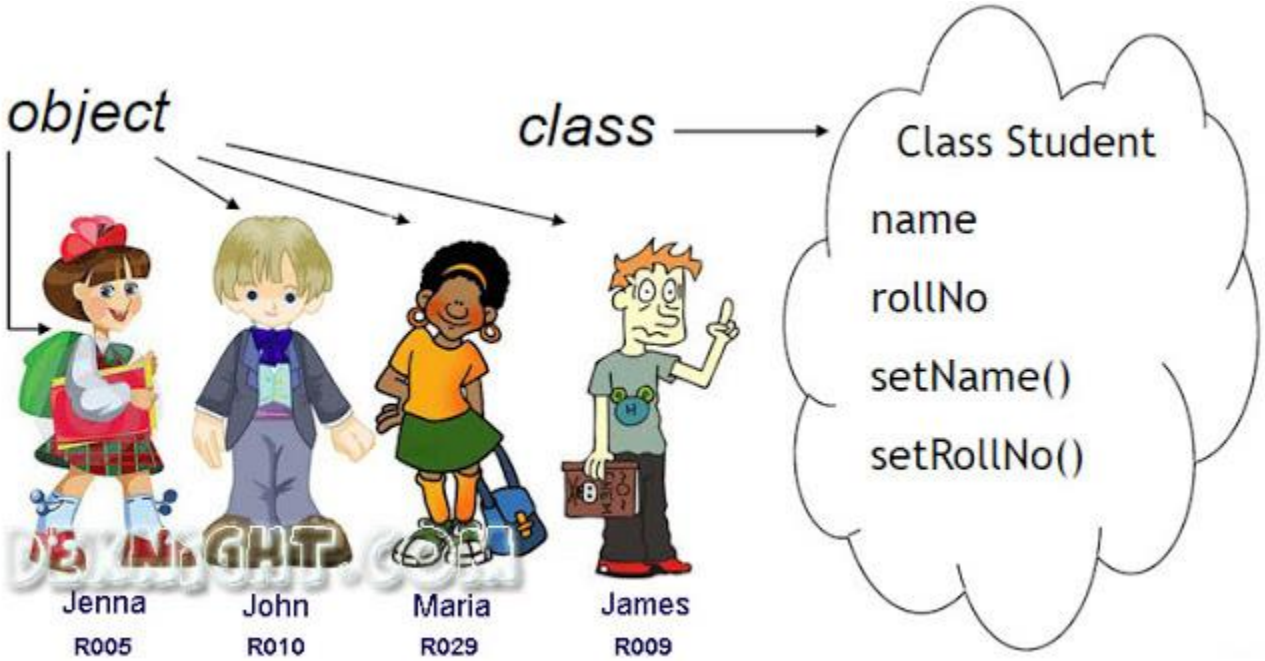
- Class and object
- Writing class
- Access specifier
- Accessor and mutator methods
- Constructor
- Concept of ‘this’ reference
- Static member and static method
- Package
- Method Overloading

Class

- Class can be defined as
  - It is **User Defined Data type (UDT)**
  - Template for creating like objects
  - Collection of data members and member functions.

Class and object





Abstraction

- Process of identifying key aspects of an entity and ignoring the rest.

- Types
  - Data Abstraction
    - empId
    - empName
    - basicSalary
  - Behavioral Abstraction
    - displayData
    - calculateSalary

Encapsulation

- Process of hiding the data of object from outside world.
- Interaction with data will be done only with the help of interface / behavior / method

Defining class

- Syntax

```
class <class name>
{
  <access specifier><datatype><member>;
  <access specifier><datatype><member>;
  <access specifier><datatype><member>;
  <access><return type><Method
  name>(<parameters>)
  {
     Statement(s);
  }
}
```

Class

```
class MyDate
{
  private int day;
  private int month;
  private int year;

  public void displayDate()
  {
     Sysout(“Date is: “ + day + “-” +
  month + ”-” + year);
  }
}
```

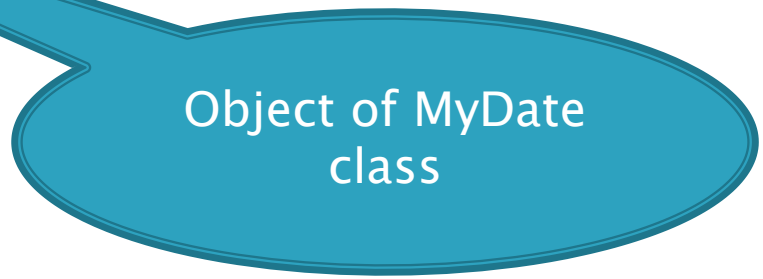
Access specifier

- Specifies accessibility of a member of class.
- Java supports 4 access specifiers.
  - **public** – accessible outside the class only with the help of object of class.
  - **default** – it is the default access specifier. It is like a public but it is not accessible outside the package.
  - **private** – Not accessible outside of class
  - **protected** – Accessible in derived class

Object

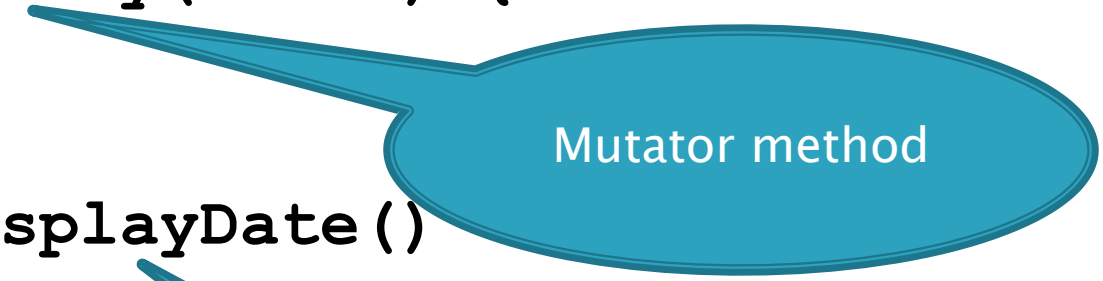
- Object is an instance of class.

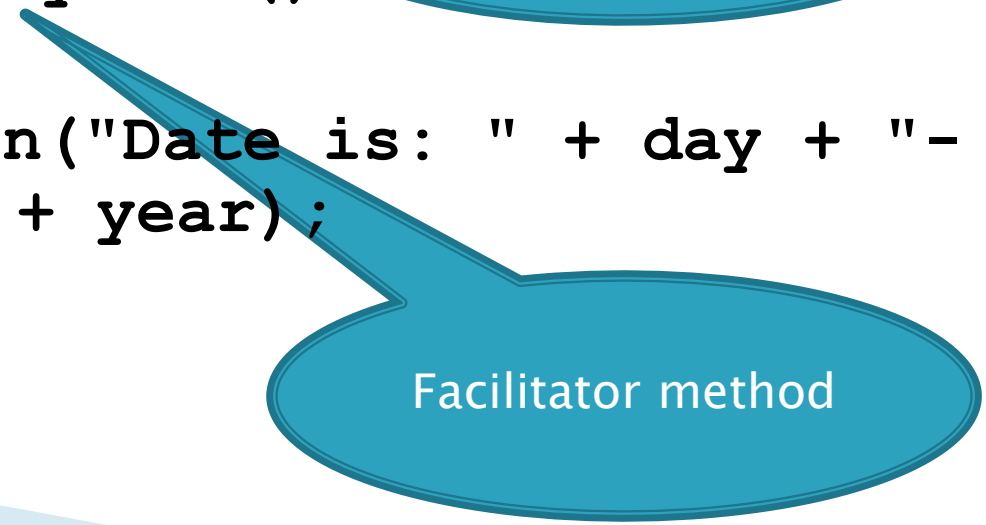
```
public static void main(String[] args)
  {
  MyDate d1 = new MyDate();
  d1.displayDate();
}
```



Accessor and Mutator methods.

```
public int getDay() {
  return day;
}
public void setDay(int d) {
  day = d;
}
public void displayDate()
{
System.out.println("Date is: " + day + "-
  " + month + "-" + year);
}
```





Constructor

- Constructor is a special member function of class.
- It has same name as class.
- No return type (not even void)
- Invoked implicitly when object is created.
- Used to initialize data members of class.
- Can be overloaded. (Class can have more than one constructor)

Constructor

```
public MyDate() {
day = 1;
month = 1;
year = 2018;
}
public MyDate(int d, int m, int y) {
day = d;
month = m;
year = y;
}
```

Plain / Default / No parameter constructor

Parameterized constructor

‘this’ reference

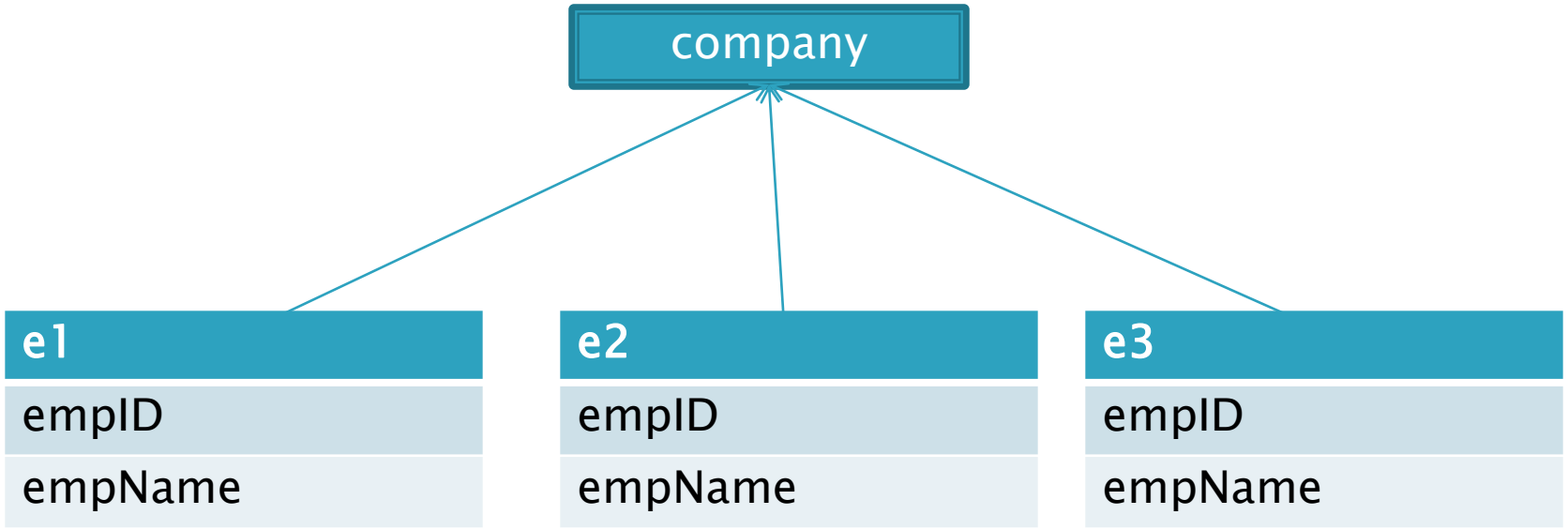
- Every **class method** is having an implicit hidden parameter a ‘this’ reference.
- ‘this’ holds address of an object which is invoking the method.

- Remove shadow of the instance field.
- Used to access current object in the method.

Static Member

- When any member of class needs to be common for all objects, such data members are declared as static.
- Such members having single copy per class. (All objects will having the same copy)
- Where instance members (non-static) having separate copy per object.

Static member



Static method

- Static methods are invoked by class name.
- Reference ‘this’ is never passed to static method. (Because those are called by class name/ not called by an object)
- Non static members (instance members) can’t be accessed in static methods.

public static void main()

- Why main() is declared as static?
  - main() is declares as static because it is invoked before instantiation of class.
  - If it is not declared as static then OS has to create object of class first and then it needs to be called.
  - So main() is declared as static.

Method overloading

- Method overloading is the concept of methods having same name but different signature.
- Function signature consists of
  - Number of parameters passed to function
  - Datatypes of parameters passed to function
  - Sequence of parameters passed to function.
- add(int, int)
- add(int, int, int)
- add(double, double)
- add(int, double)
- add(double, int)


Thank
You!