Constructor and Overloading

Sungchul Lee

Learning Object

- Constructor
  - Initialize Object Variable
- this keyword
- Overloading Constructor

Initialize Object Variable

- What if we want to initialize the Object Variable when generate/declare object?
  - Constructor !!
- A constructor in Java is a special method that is used to initialize objects
  - Reduce errors
  - Fixed format

```java
//Animal.java
public class Animal {
    public String name;
}
```

```java
//Main.java
public class Main {
    public static void main(String[] args) {
        Animal cat = new Animal();
        System.out.println(cat.name); // null Error
    }
}
```

Constructor

- Syntax
  - class **ClassName**
    { modifier **ClassName** (parameter) { statements; } }
- Class name and Constructor name must be same
- Special **method**
  - **Not required return type**
- All class have at least one constructor – default Constructor
  - if you don’t make constructor in the class
    ClassName (){ }   // is automatically created in the class
  - If you make a constructor, default constructor is not genertated

Constructor - Example

➢Initialize Instance/Object variable: name

```
//Animal.java
public class Animal {
    public String name;
    public Animal ( String name){
        this.setName(name)
    }
    public void setName( String name){
        this.name = name;
    }
}
```

```
//Main.java
public class Main {
    public static void main(String[] args) {
        Animal cat = new Animal(“Pcat”);
        System.out.println(cat.name); //Pcat
    }
}
```

this keyword

- this **keyword** in **java** can be used inside the Method or constructor of Class.
- It(this) works as a reference to the current Object, whose Method or constructor is being invoked.
  - Constructor, methods, variables
- This **keyword** can be used to refer to any member of the current object from within an instance Method or a constructor

this keyword – Examples

1. Reference current object variable
   - ❑this.variable
     - ❖this.name
2. Reference current object Constructo
   - ❑this(Constructor’s params)
     - ❖this(name)
3. Reference current
   - ❑this.method(params)
     - ❖this.setName(Name)

```
//Animal.java
public class Animal {
    public String name;
    public Animal (){
        this(“Jone Doe”)
    }
    public Animal ( String name){
        this.setName(name)
    }
    public void setName( String
    name){
        this.name = name;
    }
}
```

Overloading Constructor

- Why we need the constructor?
  - Initializing necessary elements (variables)
  - **Overloading**
    - Save time to make class
- Class can have multiple constructors
  - More than one
  - Easy to create various object without extra coding and class
- Performed a different constructor based on the parameters
  - Not allowed same number of parameter and type orders
  - e.g.

    ```
    Character(){}
    Character(String name){}
    ```

Overloading - Example

```
public class Animal {
    public String name;
    public Animal(){
        this.setName(“John Doe”);
    }
    public Animal ( String name){
        this.setName(name);
    }
    public void setName( String name){
        this.name = name;
    }
```

```
//Main.java
public class Main {
  public static void main(String[] args) {
    Animal cat = new Animal(“Pcat”);
    Animal dog = new Animal();
    System.out.println(cat.name); //Pcat
    System.out.println(dog.name);// John Doe
  }
}
```

Practice

1. Make a new project (Reference: Create Project and Class File)
    - ❑Project name: Constructor_Overloading
2. Create two Class Files
    - ❑Class name: Animal
    - ❑Class name: Main
3. Coding:

Practice - Coding

```
//Animal.java
public class Animal {
public String name; // Object Variable
public int age;

public Animal(){ // no paramter
	this.setName("Jone Doe");
	this.setAge(0);
}
public Animal(String name){//One string param
	this.setName(name);
	this.setAge(0);
}
public Animal(int age){//One int type param
	this.setName("Jone Doe");
	this.setAge(age);
}
```

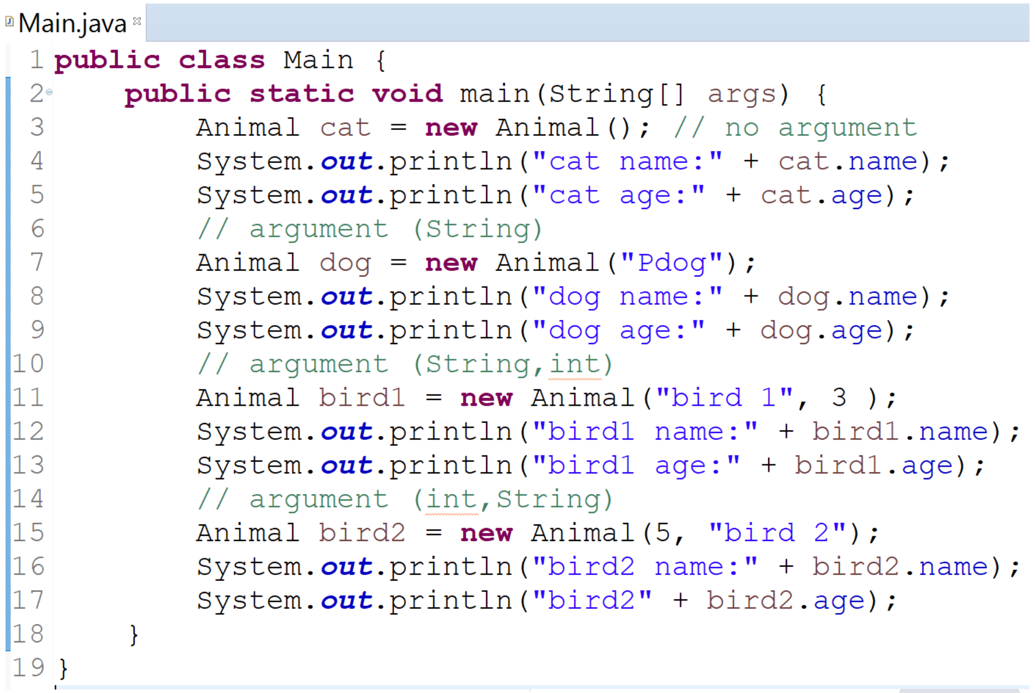
```
// one String type and one int type param
	public Animal(String name, int age){
		this.setName(name);
		this.setAge(age);
	}
// one int type and one String type param
	public Animal(int age, String name){
		this.setName(name);
		this.setAge(age);
	}
	public void setName(String name) {
		this.name = name;
	}
	public void setAge(int age) {
		this.age = age;
	}
}
```

Practice - Main

```
 //Main.java
public class Main {

public static void main(String[] args) {
// TODO Auto-generated method stub
	Animal cat = new Animal(); // no parameter
	System.out.println("cat name:" + cat.name);
	System.out.println("cat age:" + cat.age);

	Animal dog = new Animal("Pdog"); // String
	System.out.println("dog name:" + dog.name);
	System.out.println("dog age:" + dog.age);
```

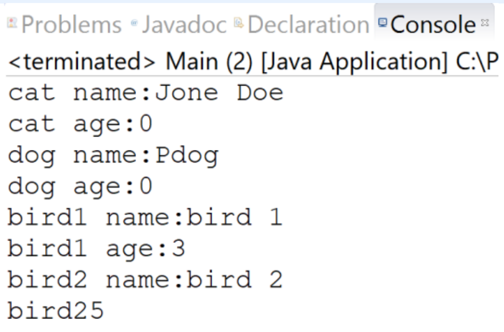
```
	Animal bird1 = new Animal("bird 1", 3 ); // String , int
	System.out.println("bird1 name:" + bird1.name);
	System.out.println("bird1 age:" + bird1.age);

	Animal bird2 = new Animal(5, "bird 2"); //int, string
	System.out.println("bird2 name:" + bird2.name);
	System.out.println("bird2" + bird2.age);

	}
}
```

Practice – Code and Result

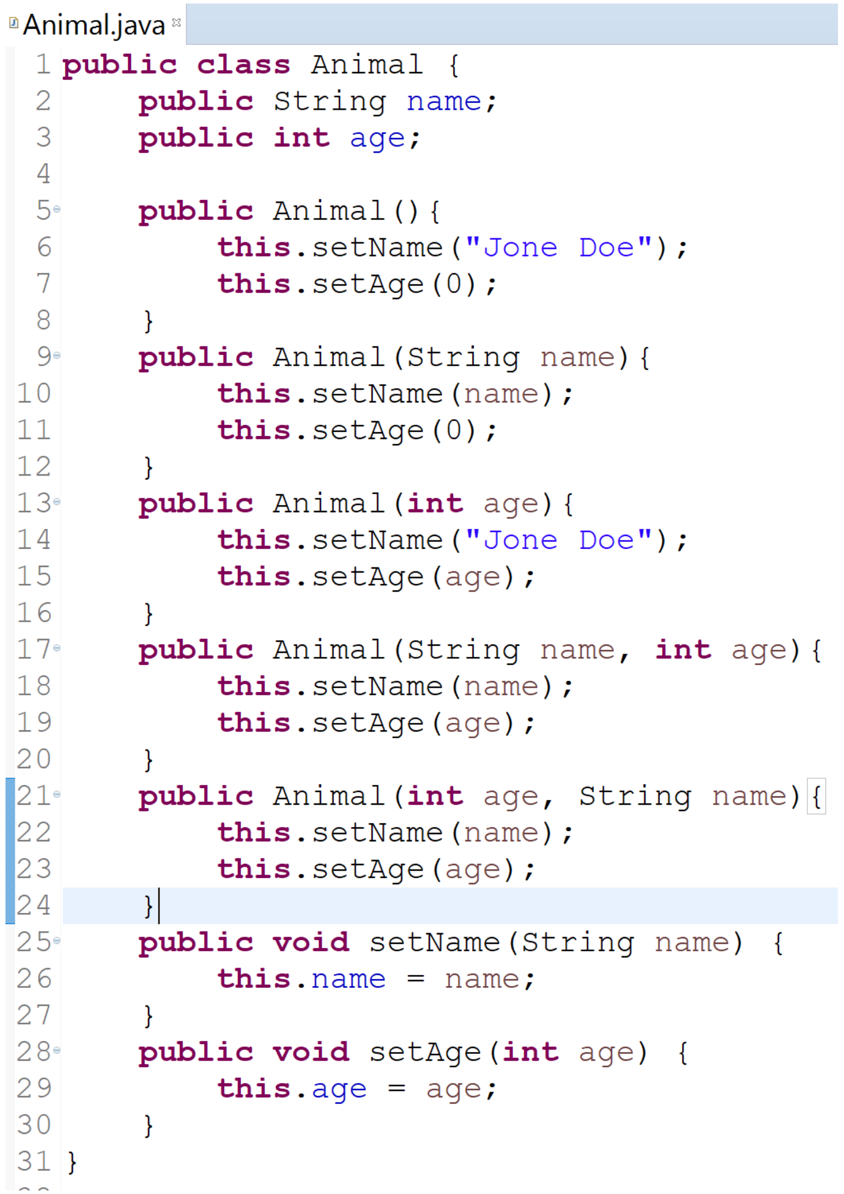


```java
public class Main {
    public static void main(String[] args) {
        Animal cat = new Animal(); // no argument
        System.out.println("cat name:" + cat.name);
        System.out.println("cat age:" + cat.age);
        // argument (String)
        Animal dog = new Animal("Pdog");
        System.out.println("dog name:" + dog.name);
        System.out.println("dog age:" + dog.age);
        // argument (String,int)
        Animal bird1 = new Animal("bird 1", 3 );
        System.out.println("bird1 name:" + bird1.name);
        System.out.println("bird1 age:" + bird1.age);
        // argument (int,String)
        Animal bird2 = new Animal(5, "bird 2");
        System.out.println("bird2 name:" + bird2.name);
        System.out.println("bird2" + bird2.age);
    }
}
```

Result



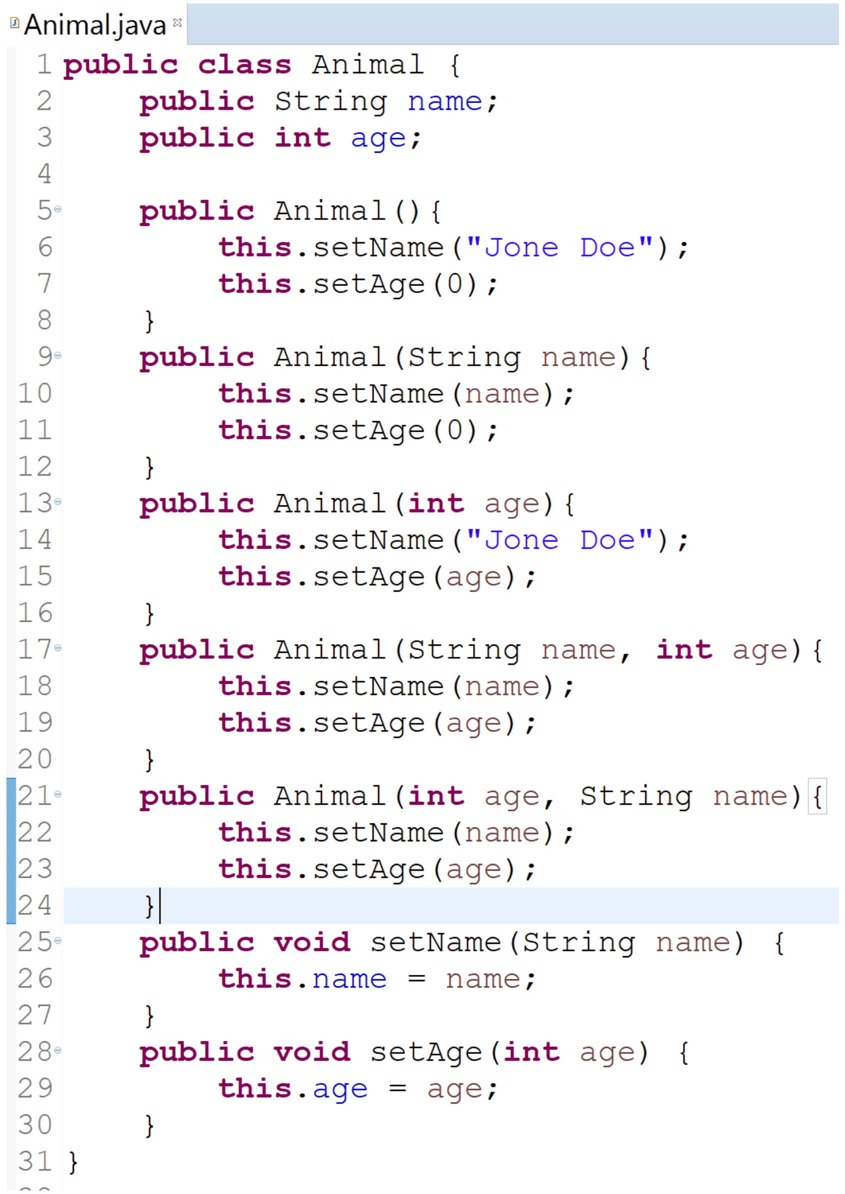
```
<terminated> Main (2) [Java Application] C:\P
cat name:Jone Doe
cat age:0
dog name:Pdog
dog age:0
bird1 name:bird 1
bird1 age:3
bird2 name:bird 2
bird25
```



```java
public class Animal {
    public String name;
    public int age;

    public Animal(){
        this.setName("Jone Doe");
        this.setAge(0);
    }
    public Animal(String name){
        this.setName(name);
        this.setAge(0);
    }
    public Animal(int age){
        this.setName("Jone Doe");
        this.setAge(age);
    }
    public Animal(String name, int age){
        this.setName(name);
        this.setAge(age);
    }
    public Animal(int age, String name){
        this.setName(name);
        this.setAge(age);
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

Summary

- Constructor
  - Initialize Object variables
  - Same name as class name
  - No return type
  - More than one constructor
    - Default constructor ClassName(){}
- Overloading
  - Distinguished by
  - number of parameters
  - Order of parameter’s type


Animal.java

```
public class Animal {
    public String name;
    public int age;

    public Animal(){
        this.setName("Jone Doe");
        this.setAge(0);
    }
    public Animal(String name){
        this.setName(name);
        this.setAge(0);
    }
    public Animal(int age){
        this.setName("Jone Doe");
        this.setAge(age);
    }
    public Animal(String name, int age){
        this.setName(name);
        this.setAge(age);
    }
    public Animal(int age, String name){
        this.setName(name);
        this.setAge(age);
    }
    public void setName(String name)  {
        this.name = name;
    }
    public void setAge(int age)  {
        this.age = age;
    }
}
```