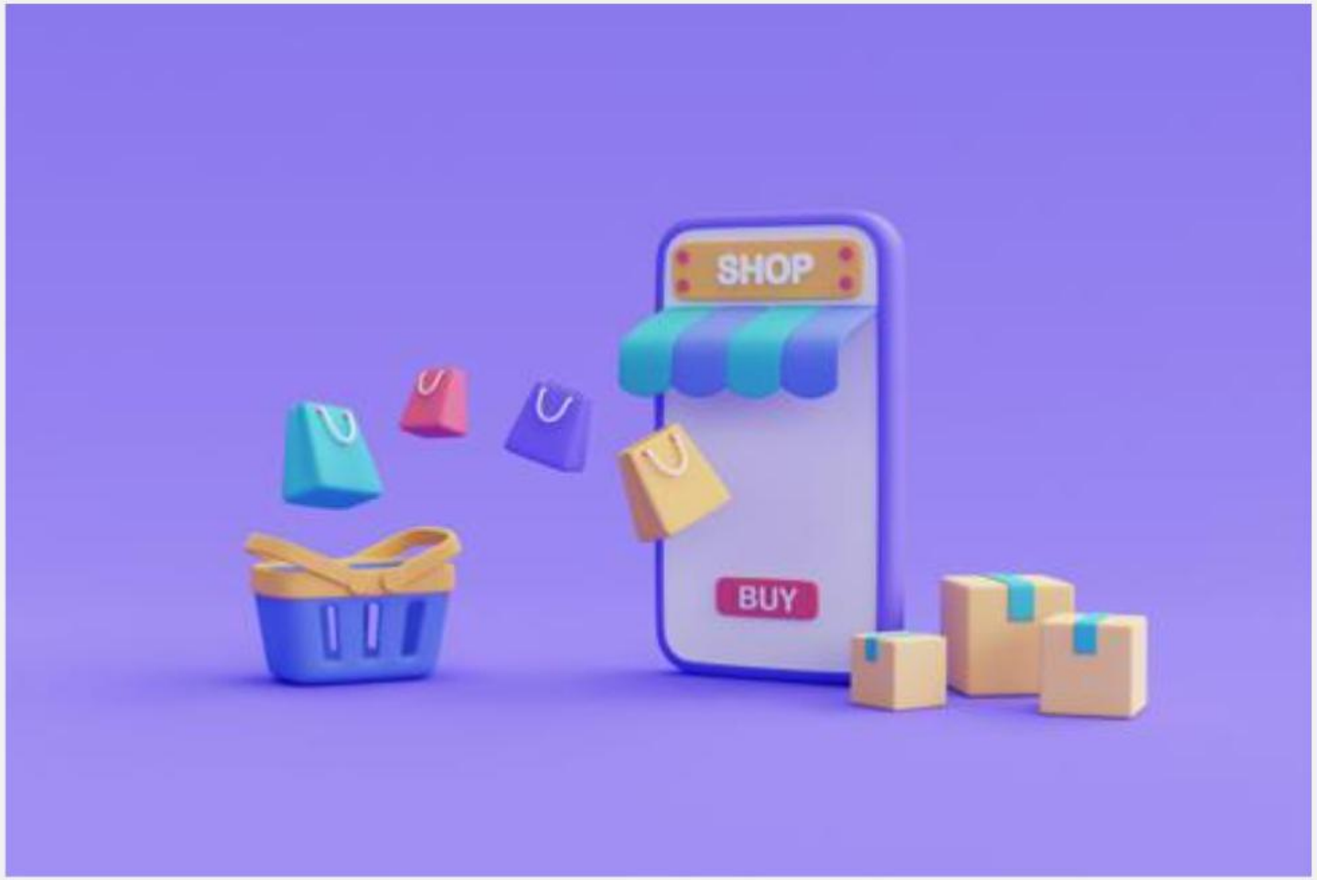
Objects in a Domain – Bank Account

- Imagine that you wish to open a bank account.
  - What information will your bank need to open your account?
  - Is bank account an object?



Objects in a Domain – Ecommerce Application

- When shopping on any ecommerce platform, what information must you provide to buy a product?
- Is the Ecommerce application an object?

Creating Objects and Constructors

Learning Objectives

- Explore Java objects
- Objects in Memory
- Define constructors
- Declare different types of constructors
- Use “this” keyword
- Explore static and final modifiers in Java

Explore Java Objects

Let's Recall

- What are objects in the real world?
- Can real world objects be modeled in Java applications?
- How did you identify the objects, their attributes, and behavior?
- What are classes?
- How are attributes modeled in a class?
- How is behavior modeled in a class?

Java Objects

- What are Java objects?
  - Java objects are the most fundamental units of the Object-oriented programming model.
  - Objects in Java consists of attributes and behavior.
    - Attributes - Determined by the properties of the Object.
    - Behavior – Determined by the way the Object reacts or acts in multiple scenarios.

```java
public class Employee {
    String employeeName;
    int employeeCode;
    int age;
    String dob;
    double salary;

void displayEmployeeDetails(){
    System.out.println("The details of an employee are");
    System.out.println(employeeName+"::"+employeeCode);
    System.out.println(age+"::"+dob+"::"+salary);
}
```

Creating Objects in Java

- Objects in Java can be created from the class.
- Consider the class Employee with the attributes `employeeName, employeeCode,age,dob,salary.`
- The class contains the blueprint, or the template needed to create an object.
- The object created will consist of data in the form of variables and code in the form of methods.

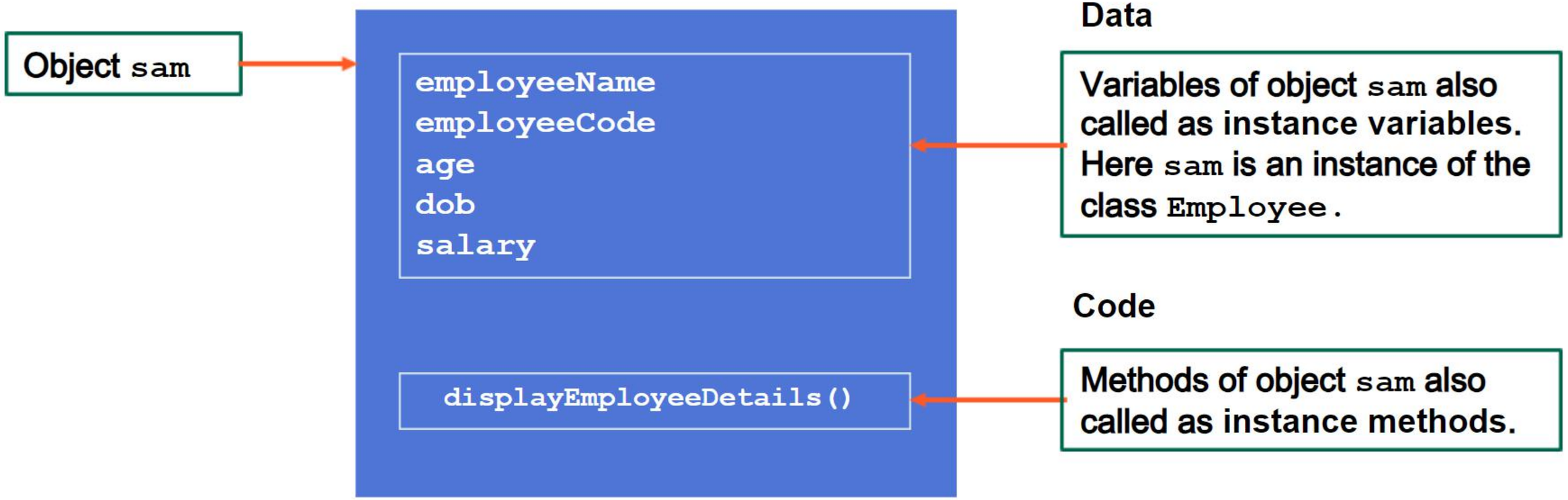
Creating Objects in Java (contd.)

- The `new` keyword is used to create the object of Employee.
- The object with the name `sam` is created in the main method of the class.
- The object `sam` will have attributes that are unique to `sam`.

```
public class Employee {
    String employeeName;
    int employeeCode;
    int age;
    String dob;
    double salary;

    void displayEmployeeDetails(){
        System.out.println("The details of an employee are");
        System.out.println(employeeName+"::"+employeeCode);
        System.out.println(age+"::"+dob+"::"+salary);
    }

    public static void main(String[] args) {
        Employee sam = new Employee();
    }

}
```

Instance Variables and Methods

- The object created will consist of the following:



- The instance variables and methods belong to the specific object. Once the object is deleted or goes out of scope the values of the variables will be reset.
- Instance variables are given a default value before they are initialized.

Java Class and Objects

- Objects in Java can be created from the class.
- Consider the class Employee with the attributes `employeeName, employeeCode,age,dob,salary.`
- Each employee of the organization will have different attributes that will uniquely identify the employee.
- Similarly, multiple objects with different attribute values can be created.

```
public class Employee {
    String employeeName;
    int employeeCode;
    int age;
    String dob;
    double salary;
}
```

Martin
E1009
22
09/08/2000
$2000

Belle
E1045
24
01/03/1998
$1500

Anne
E1209
42
01/03/1981
$4000

Gunther
E1100
36
09/08/1986
$3500

Object Declaration and Initialization

- Declaring an object creates a variable that will hold the reference to the object.
- When you declare an object, memory is not allocated to it.
- To allocate memory to the object, you need to instantiate or initialize the object by using the `new` keyword.
- The object can be declared first and initialized later.

```
Employee sam;
sam = new Employee();
```

- It can also be declared and initialized at the same time.

```
Employee sam = new Employee();
```

- Multiple objects can be created for the class as follows:

```
Employee anne = new Employee();
Employee belle = new Employee();
Employee martin = new Employee();
```

Instance Variables – Default Values

- Once an object is created using the new keyword, memory is allocated for the object's instance variables and methods.
- The instance variables are given some default values.
- Primitive types like `int,float, double,byte,short,long` will be initialized to `0`.
- `char` will be initialized to `'/u000'`
- `boolean` will be initialized to be `false`.
- Non-primitive datatypes like String and arrays are initialized to `null`.

```java
public class Employee {
    String employeeName;
    int employeeCode;
    int age;
    String dob;
    double salary;

    void displayEmployeeDetails(){
        System.out.println("The details of an employee are");
        System.out.println(employeeName+"::"+employeeCode);
        System.out.println(age+"::"+dob+"::"+salary);
    }

    public static void main(String[] args) {
        Employee sam = new Employee();
    }

}
```

Accessing Variables and Methods of a Class

```
Employee sam = new Employee();
System.out.println(sam.employeeName);
System.out.println(sam.employeeCode);
```

The “.” operator is used to access variables and methods of a class

Assign Values to Instance Variables

- The objects are created but the instance variables have the default values only.
- Values can be assigned to the variables of the object by accessing the variable using the "." dot operator.

```
<object-name>.<variable-name> = <value>
```

- Assign values to the variables of the object `'anne'`.

```
Employee anne = new Employee();
anne.employeeCode = "E1209";
anne.employeeName = "Anne J";
anne.age = 42;
anne.dob = "09/08/2000";
anne.salary = 4000;
```

- Assign values to the variables of the object `'belle'`.

```
Employee belle = new Employee();
belle.employeeCode = "E1045";
belle.employeeName = "Belle";
belle.age = 24;
belle.dob = "01/03/1998";
belle.salary = 1500;
```

Note that `anne` and `belle` have different values assigned to the variables.

Library

Sam, a librarian in a college, is struggling to get his books organized.

The library has books on various courses they offer. As part of an initiative to reorganize the library and make it more accessible for the students, the college has decided to create a library management system to provide an online platform to search for and access various books and materials. Sam, who is part of this initiative has to create a Book class that will help the system organize the books.

Help Sam create the class and capture minimum attributes like title, year of publishing, ISBN Number, and author name.

Click here for the solution.
Use the IntelliJ IDE for the demonstration.

DEMO

Objects in Memory

Objects in the Memory

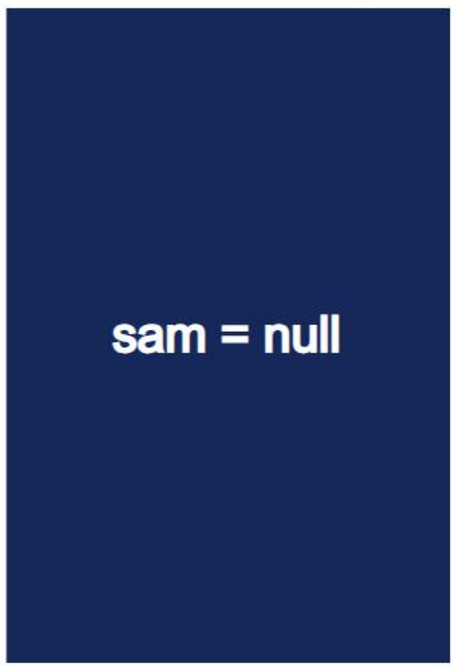
- There are two types of memory that the JVM designates for the object.
  - Stack memory – The objects declared, are created as reference variables in the stack memory, which references to null.

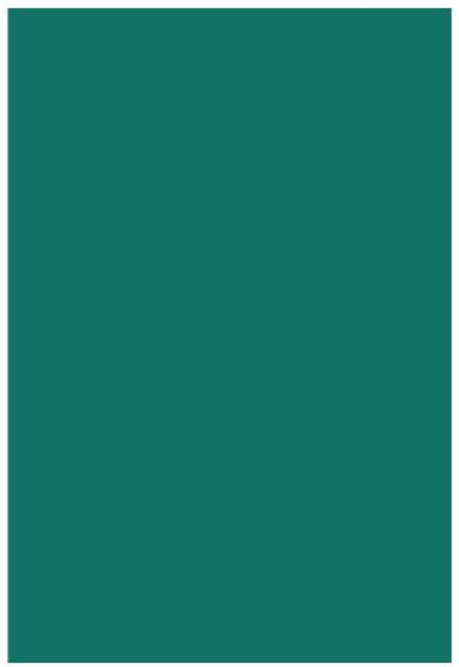
```
Employee sam;
```

  - Heap Memory – The objects that are initialized using the `new` keyword are:
    - Allocated space in the heap memory.
    - And the reference variable in the stack holds the memory address of the heap instead of null after initialization.

```
sam = new Employee();
```

Stack and Heap Memory – Object Declaration



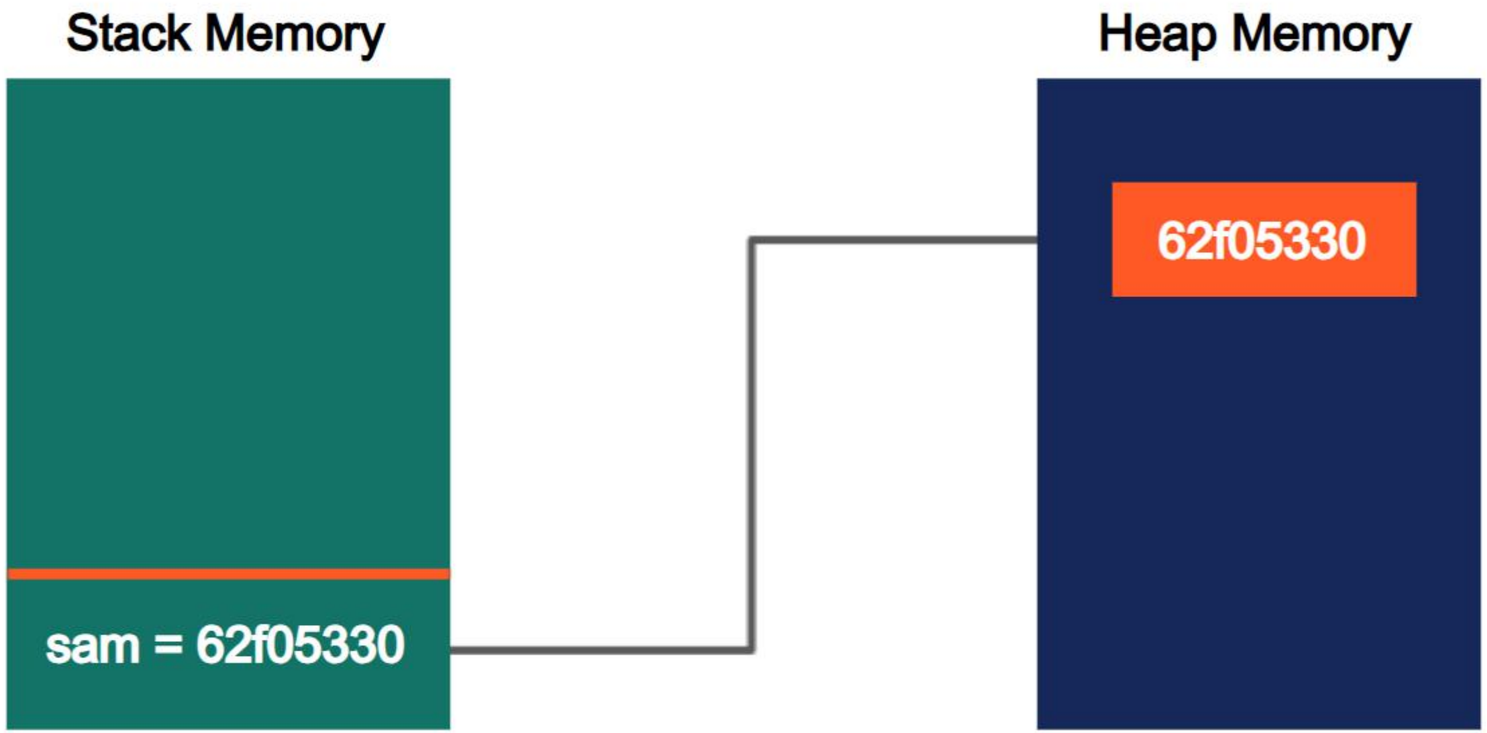


Object Declaration

```
Employee sam;
```

- An object `sam` of `Employee` type is declared in the stack but it does not hold any values and is empty or null.
- Here `sam` is the reference variable.

Stack and Heap Memory – Object Initialization



Object Initialization

```
sam = new Employee();
```

- The reference variable `sam` created in the stack memory will point to the actual memory space allocated in the heap after the object `sam` is initialized using the `new` keyword.

Quick Check

The _____ key word is used to initialize an object.

1. new
2. void
3. class
4. int

Quick Check: Solution

The _____ key word is used to initialize an object.

1. `new`
2. `void`
3. `class`
4. `int`

Define Constructors

What Is a Constructor?

- A constructor is a special method of the class that initializes a newly created object.
- Constructor declarations are similar to method declarations - except that they use the name of the class and have no return type.
- A constructor is invoked to create objects from the class blueprint when the `new` keyword is used.

```
Employee(){

}
```

A constructor

The `new` Keyword

- What happens when the `new` keyword is used?
  - The compiler internally invokes a default constructor.
  - This default constructor is provided by the JVM for every class once the object of the class is created using the `new` keyword.
  - Note that the default constructor is provided only to those classes that do not have a constructor defined within them.

```
public static void main(String[] args) {
    Employee sam = new Employee();
}
```

Important Points to Remember – Constructors

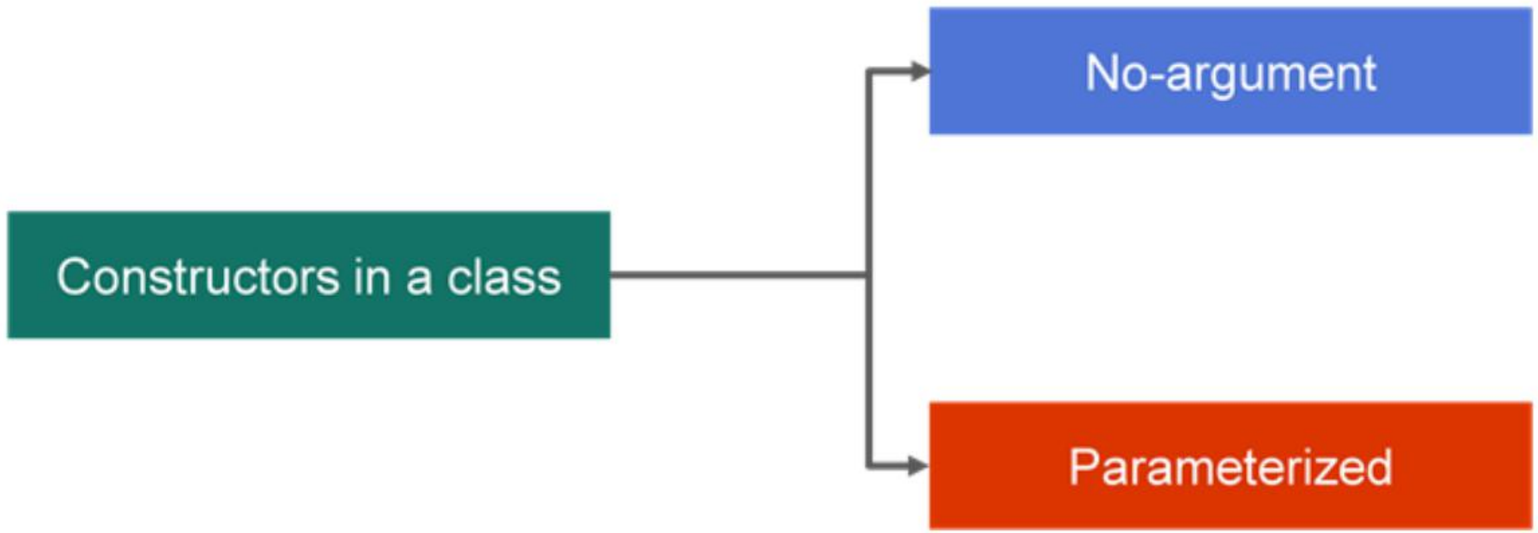
- The constructor must have the same name as the class name.
- The constructors are called only once during object initialization.
- They cannot be called explicitly like methods using the dot operator.
- The variables declared in the class can be given some initial values inside the constructor.

Declare the Different Types of Constructors

Types of Constructors in a Class

There are two types of constructors that can be defined inside a class:

1. No-argument constructor
2. Parameterized constructor



```
Employee(){
    employeeName = "Sam";
    employeeCode = 102;
    age = 24;
    dob = "03/03/1998";
    salary = 1000;
}
```

A no-argument constructor

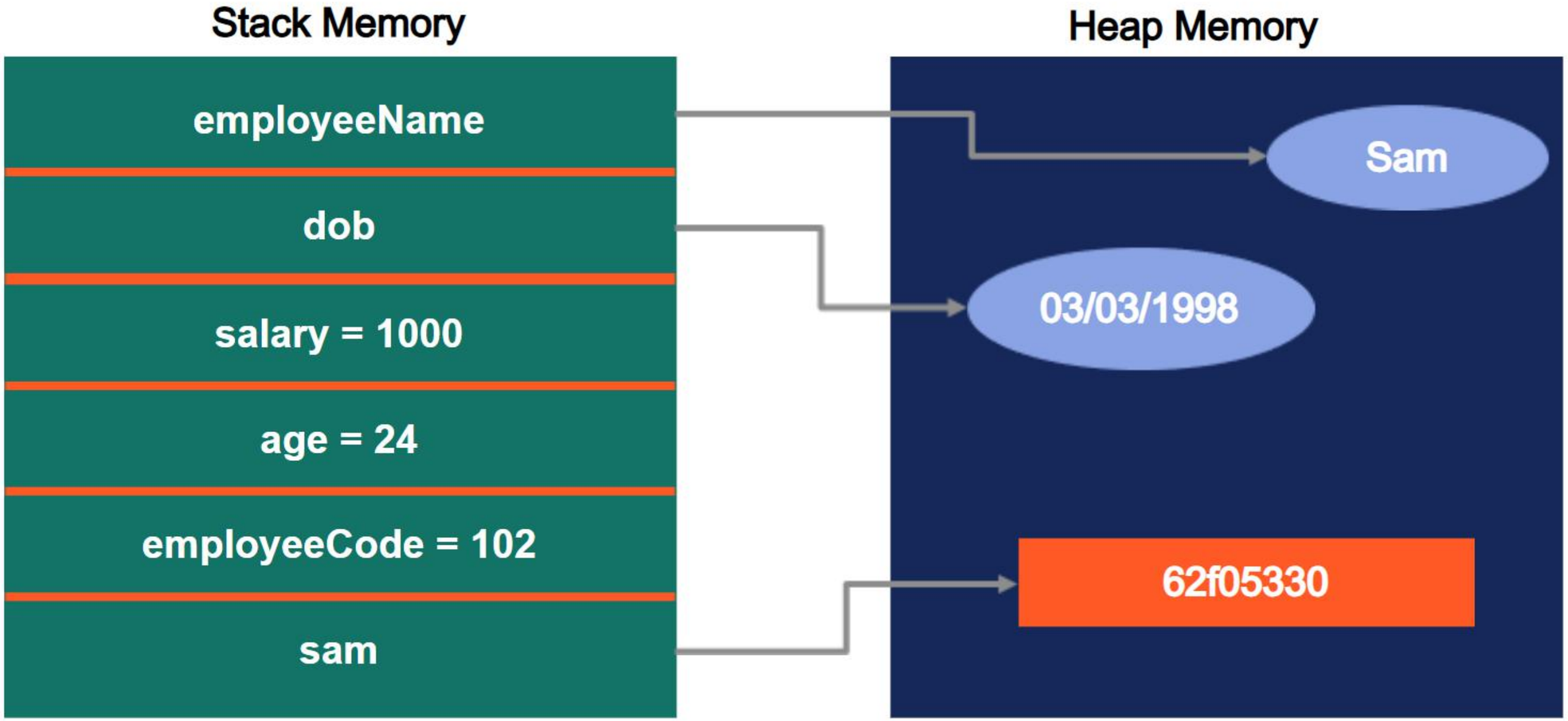
```
Employee sam = new Employee();
```

Instantiating an object that invokes the no-argument constructor

No-argument Constructor

- If a constructor does not accept any parameters and is defined in the class, it is known as a no-argument constructor.
- The attribute values can be initialized inside constructors since constructors are called only during the initialization.
- Instead of default values the instance variables will hold the values initialized in the constructor.
- For every object created, the attributes of the object will be the same.

Stack and Heap Memory – Object Initialization



- The objects will have values stored in heap memory and reference to them.
- For the primitive datatypes like int, float, double etc., the values will be stored in the stack memory itself.

Library

Sam, a librarian in a college, is struggling to get his books organized.

The college library contains books on the various courses they offer. As part of an initiative to reorganize the library and make it more accessible for the students, the college has decided to create a library management system to provide an online platform to search for and access various books and materials. As a part of this, Sam must create a Book class that will help the system organize the books.

Help Sam create the class and capture minimum attributes like title, year of publishing, ISBN Number, and author name.

1. Create an object for the Book class in an implementation class BookImpl.java
2. Initialize the variables in the no-argument constructor
3. Display the details.

Click here for the solution.

Use the IntelliJ IDE for the demonstration.

DEMO

Parameterized Constructor

- A Java constructor can also accept one or more parameters. Such constructors are known as parameterized constructors.
- The parameters are passed to the constructor at the time of initialization.
- So different objects created can have different attributes.

```
Employee(String employeeName,int employeeCode,
         int age,String dob,double salary){
    this.employeeName = employeeName;
    this.employeeCode = employeeCode;
    this.age = age;
    this.dob = dob;
    this.salary = salary;
}
```

A Parameterized Constructor

```
Employee tom = new Employee( employeeName: "Tom",
        employeeCode: 103, age: 23, dob: "08/09/1999", salary: 2000);
```

Initializing an object that invokes the parameterized constructor

Use `this` Keyword

`this` Keyword

```java
public class Employee {
    String employeeName;
    int employeeCode;
    int age;
    String dob;
    double salary;


    Employee(String employeeName, int employeeCode,
             int age, String dob, double salary) {
        this.employeeName = employeeName;
        this.employeeCode = employeeCode;
        this.age = age;
        this.dob = dob;
        this.salary = salary;
    }

}
```

Class member variable

Parameter variable

The value of the parameter variable is set to the class member variable using `this` keyword

Constructor – Without `this` Keyword

- The variables in the constructor parameter and the variables in the class have the same name.
- If `this` keyword is not used, the variables in the constructor are assigned to themselves and not to the class variables.

```
public class Employee {
    String employeeName;
    int employeeCode;
    String dob;
    int age;
    double salary;

    public Employee(String employeeName, int employeeCode,
                    String dob, int age, double salary) {
        employeeName = employeeName;
        employeeCode = employeeCode;
        dob = dob;
        age = age;
        salary = salary;
    }
}
```

Constructor – With `this` Keyword

- The most common use of `this` keyword is to eliminate the confusion between instance variables and parameter variables of the constructor with the same name.
- An instance variable is shadowed by a constructor parameter variable of the same name therefore `this` keyword is used to reference to the current object.

```
Employee(String employeeName,int employeeCode,
        int age,String dob,double salary){
    this.employeeName = employeeName;
    this.employeeCode = employeeCode;
    this.age = age;
    this.dob = dob;
    this.salary = salary;
}
```

Startup Organization

A startup company has increased its profits and decided to give a 20% pay raise to all its employee.

Tasks:

1. Create an object for Daniel with the given details.
2. Use a parametrized constructor and pass the values to it.
3. Calculate the annual salary of the employee Daniel and display the same.
4. Calculate the pay raise of Daniel and display his salary before and after the pay hike.

Click here for the solution.
Use the IntelliJ IDE for the demonstration.

Variable Name	Value
employeename	Daniel
employeecode	130
age	32
dob	12/12/1990
address	124 Bridgewater Eville
salary	$3000 a month

DEMO

Quick Check

Which of the following statement about a constructor is true?

1. A constructor can be called multiple times in a program.
2. A constructor does not have a return type.
3. A default constructor and no-argument constructor are same.
4. It is mandatory to declare a constructor in a Java program.

Quick Check: Solution

Which of the following is true about a constructor?

1. A constructor can be called multiple times in a program.
2. A constructor does not have a return type.
3. A default constructor and no-argument constructor are same.
4. It is mandatory to declare a constructor in a Java program.

Local and Instance Variables

Local Variable	Instance Variable
The variables are declared within a method but do not get any default values and must be initialized with a value before using them.	The variables are declared within a class but outside a method or constructor and always get a default value.
The scope is generally limited to a method and starts from the line they are declared. The scope ends within the block of code	The variables are created when an object of the class is created and are destroyed when the object goes out of scope.
Objects do not maintain copies of local variables.	Every object created has its own copy of the instance variables
Mandatory to provide initial values.	Default values are assigned on declaration.

```
public double calculateAnnualSalary(){
    double annualSalary = 0;
    annualSalary = salary *12;
    return annualSalary;
}
```

```
public class Employee {
    String employeeName;
    int employeeCode;
    int age;
    String dob;
    double salary;
    Address address;
```

Explore `static` and `final` Modifiers in Java

Modifiers in Java

- Modifiers are specific keywords present in Java. Using Modifiers changes can be made to the characteristics of a variable, method, or class.
- The most widely used modifiers are:
  - `final`
  - `Static`
- `final`: The `final` keyword is applicable to methods, variables, and classes. When used with a variable, the value of the variable cannot be modified or reassigned a value. Therefore, constants can be created by prefixing the variable with the `final` keyword.

```
// final variable (constant)
final double PI = 3.14159;
```

```
// re-assigning final variable,
// will throw a compile-time error
    PI = 4;
```

`static` keyword

- The `static` keyword is used to denote those class variables and methods that belong specifically to a class and not to any of its objects or instances.
- Therefore, if any method modifies the value of a `static` variable, it can be seen by all the other instances of the class.
- When an instance of a class is destroyed, the `static` variable is not destroyed and remains available to other instances of that class.
- A `static` variable or method belonging to a class can be accessed without creating an object of the class.

`static` Variables

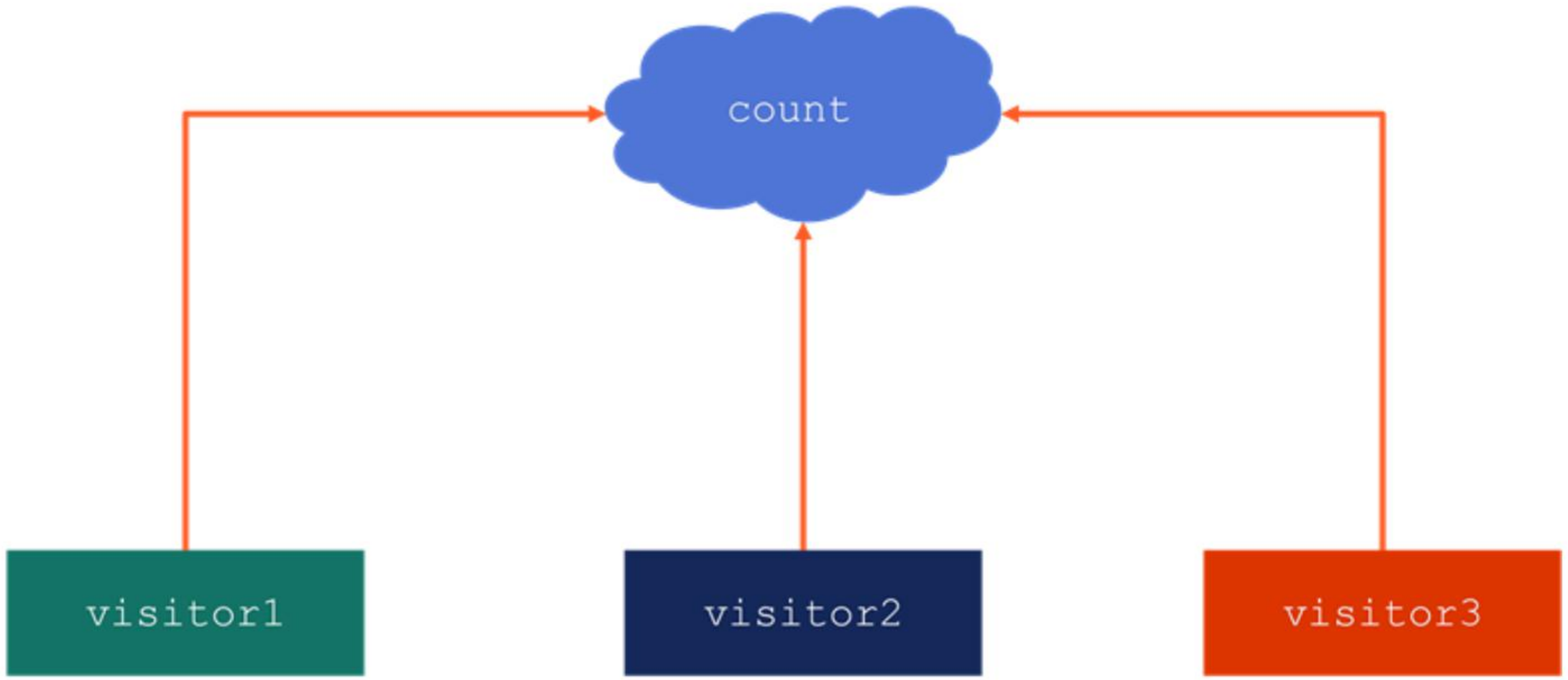
- `count` is a static variable and is incremented inside `visitorCount()` method.
- The objects of the class `visitor1, visitor2, visitor3` call on the `visitorCount()` method, yet the `count` variable is not reset each time, but maintains the value.
- Each time a call is made to the `count` variable, its value is incremented.
- A shared copy of the variable is maintained by all the objects of the class `Visitor`.

```
public class Visitor {
    static int count;
    public void visitorCount(){
        count++;
        System.out.println("You are a visitor number "+count);
    }
    public static void main(String[] args) {
        Visitor visitor1 = new Visitor();
        Visitor visitor2 = new Visitor();
        Visitor visitor3 = new Visitor();
        visitor1.visitorCount();
        visitor2.visitorCount();
        visitor3.visitorCount();
    }
}
```

```
You are a visitor number 1
You are a visitor number 2
You are a visitor number 3
```

`static` Variables and Objects

- All objects share copy of the variable count.
- If the count changes value, it is reflected to all the objects of the class.



`static` Methods

- A static method is a class method and can be invoked before the instance of a class is created.
- Writing a static method – Prefix the method with the static keyword.

```
public static double calculateAnnualSalary(double salary){
    double annualSalary = 0;
    annualSalary = salary *12;
    return annualSalary;
}
```

- Accessing a static method – Access the static method by using the class name itself.

```
public static void main(String[] args) {

    double annualSalary = Employee.calculateAnnualSalary(2000.99);
    System.out.println("The annual salary is : "+annualSalary);


}
```

The `static` `main` Method

- The `main()` method is a static method because it be invoked without creating an instance of the class to which the `main()` belongs.
- Therefore, by default the JVM expects the format of the main method to be

```
public static void main(string [] args)
```

- The `main()` thus becomes the entry point for any Java class.