Core Java part3

Trainer :Sanjay Kumar Mobile:91-9818254421, 9910587200 Email id:javaconsultant.sanjaykumar@gmail.com ,Skyp id:sanjubaba1982

Types of Control Flow Statements

- Loops
  - while loop
  - do-while loop
  - for loop

Objectives

- Iterative Statements
- while loop
- for loop
- do while loop
- Jump statement
- break jump statement
- continue jump statement

Types of Control Flow Statements

- Loops

Control Structures

Work the same as in C / C++

for, while, do/while

```
i = 0;
while(i < 10)
{
    a += i;
    i++;
}
```

```
for(i = 0; i < 10;
i++) {
    a += i;
}
```

```
i = 0;
do {
    a += i;
    i++;
} while(i <
10);
```

Control Structures (Contd…)

Java supports continue & break keywords also

Again, work very similar to as in C / C++

Switch statements require the condition variable to be a char, byte, short or int

```
for(i = 0; i< 10; i++)
{
    if(i == 5)
        continue;
    a += i;
}
```

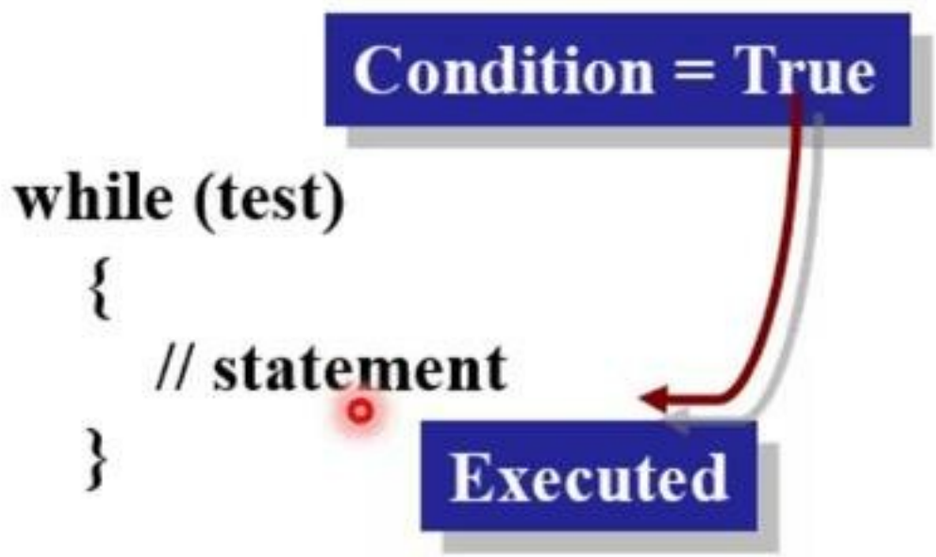
```
for(i = 0; i< 10; i++)
{
    a += i;
    if(a > 100)
        break;
}
```

Loops 3-1

while

The while loop executes a statement or set of statements as long as the condition specified evaluates to true.

Syntax



Example

```
int count = 0;
while (count < 10) {
System.out.println(count);
    count++;
}
```

Loops 3-3

for

The for loop is primarily used for executing a statement or block of statements a predetermined number of times.

Syntax

Example

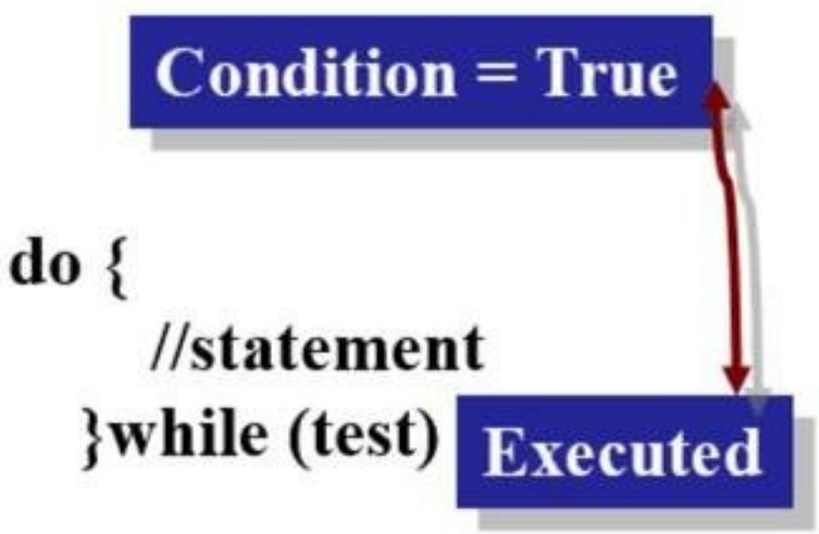
```
For(count = 0; count
<10; count++) {
System.out.println(coun
t);
}
```

Loops 3-2

do-while

The do-while loops execute certain statements till the specified condition is true. This loop ensures that the loop body is executed at least once.

Syntax



Example

```
do {
System.out.println(count);
   count++;
} while (count < 10)
```

Jump Statements 2-2

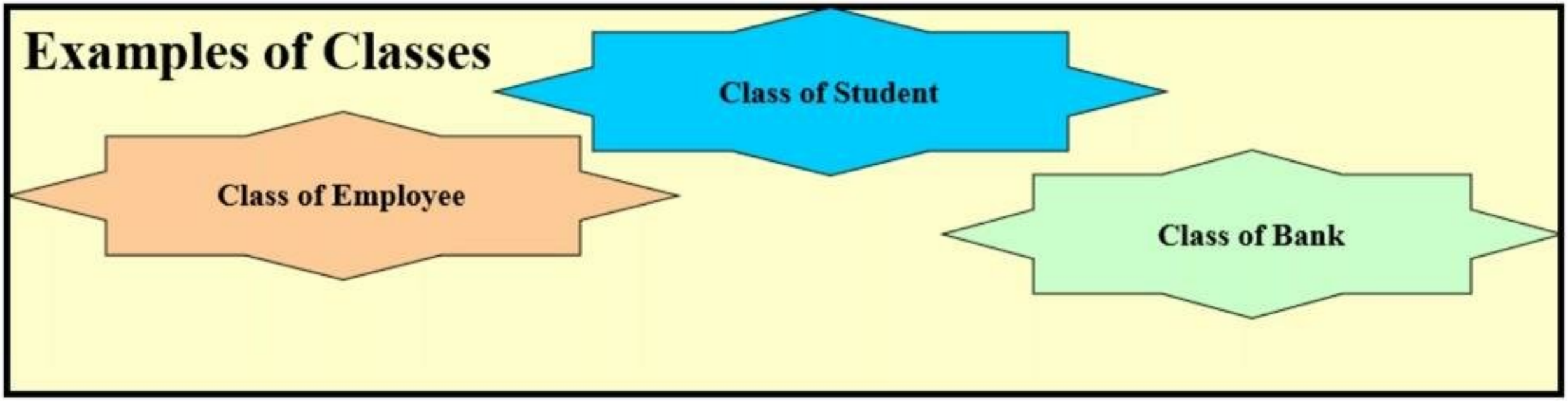
```
int number = 41;
   int is_prime = 1;

   for(int i = 2; i * i <= number; i++) {
       if((number%i)==0){
           is_prime = 0;
           break;
       }
    }
   if(is_prime==1) {
        System.out.println("It is a prime number");
   }
   else {
       System.out.println("Not a prime number");
    }
}
```

Objectives

- What is class?
- Properties and method of class
- What is an object?
- Example of objects
- Access Scope of properties and methods
- public, protected, default and private
- Method Modifier
- static, final and this

Class

- A Class defines an entity in terms of common characteristics and actions.
- Class is a mechanism used to group properties of actions common to various objects.



"A class is a blueprint for a group of objects that have common properties and behavior."

Methods

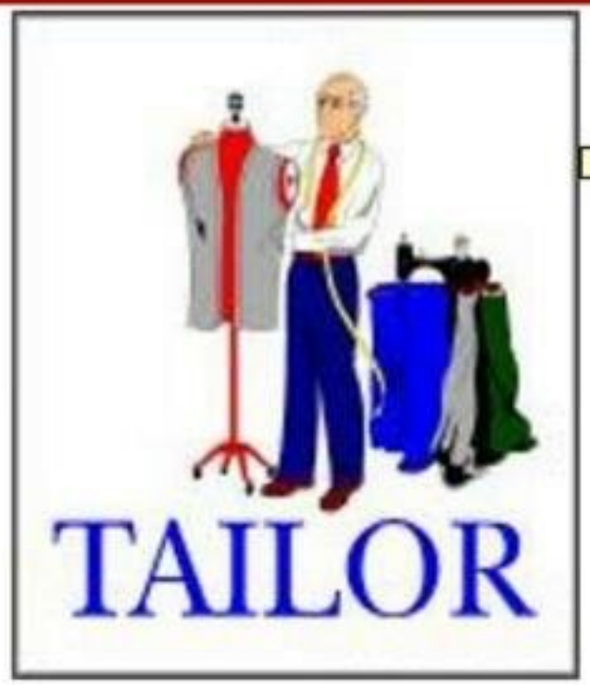
Methods

Actual implementation of an operation

Methods specify the manner in which the data of an object is manipulated

Specification of how the requested operation is carried out

Example of Method



stitchClothes

Method

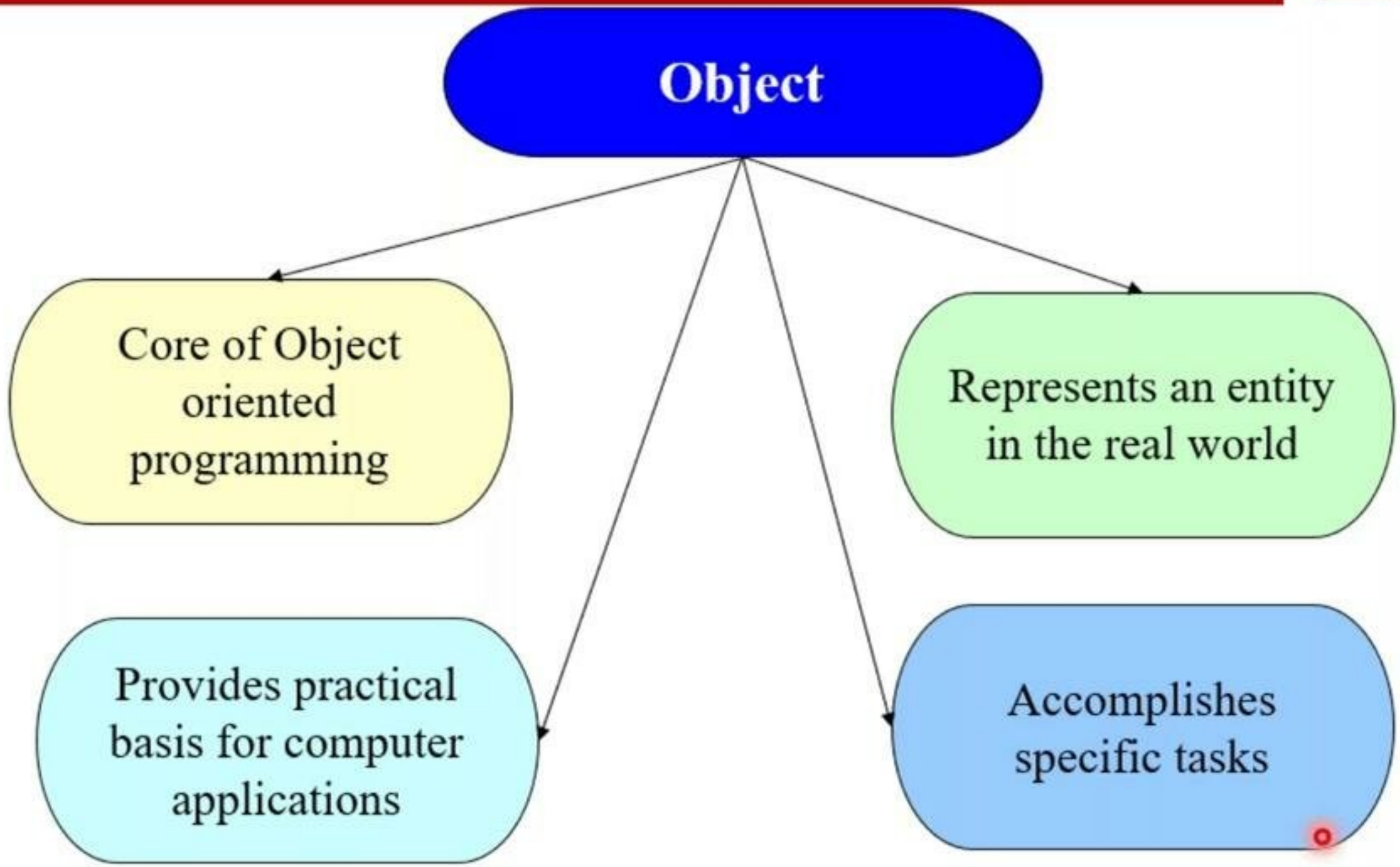
Take Measurement

Get Instruments

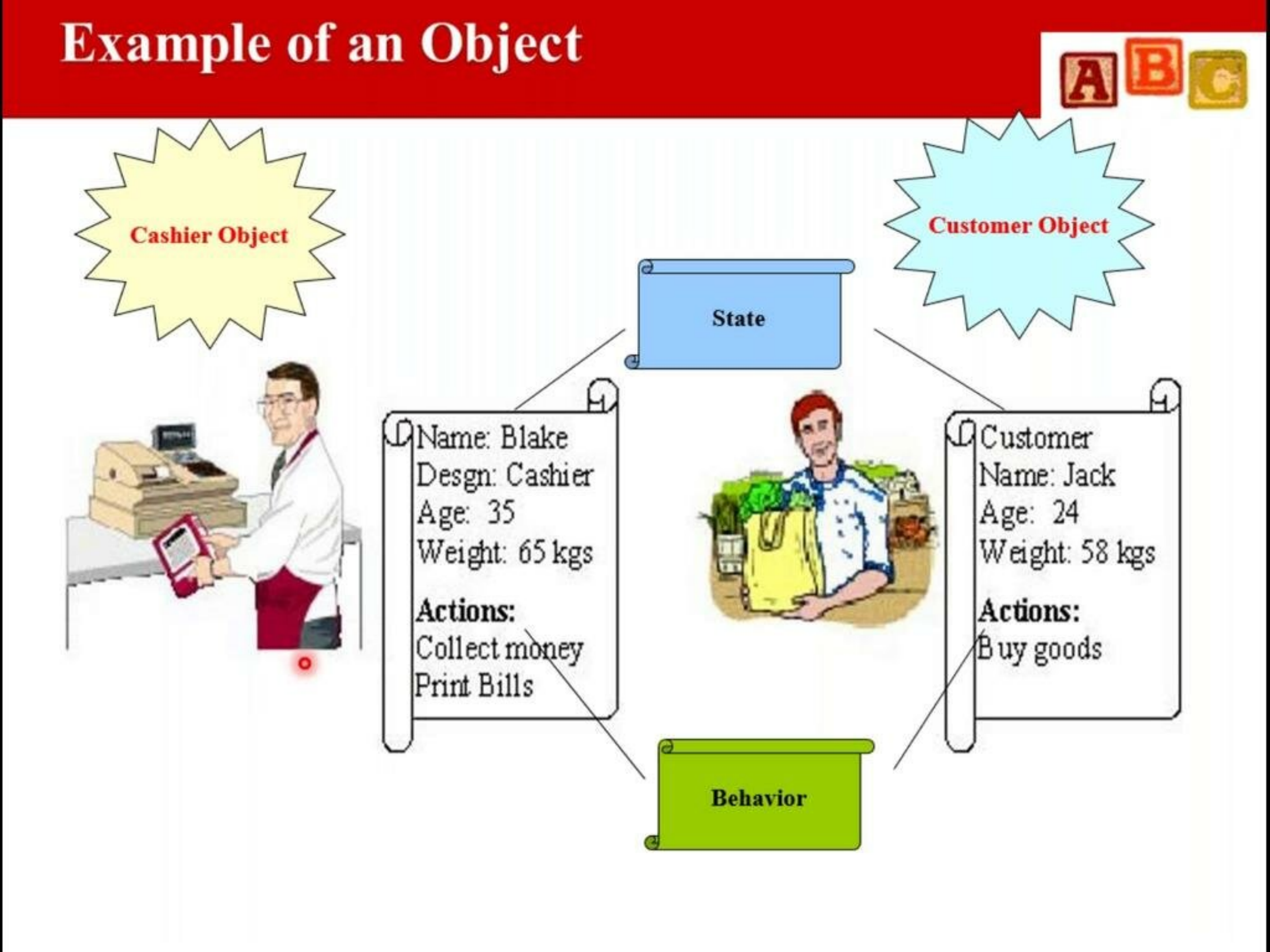
Stitch clothes

Steps for stitching clothes

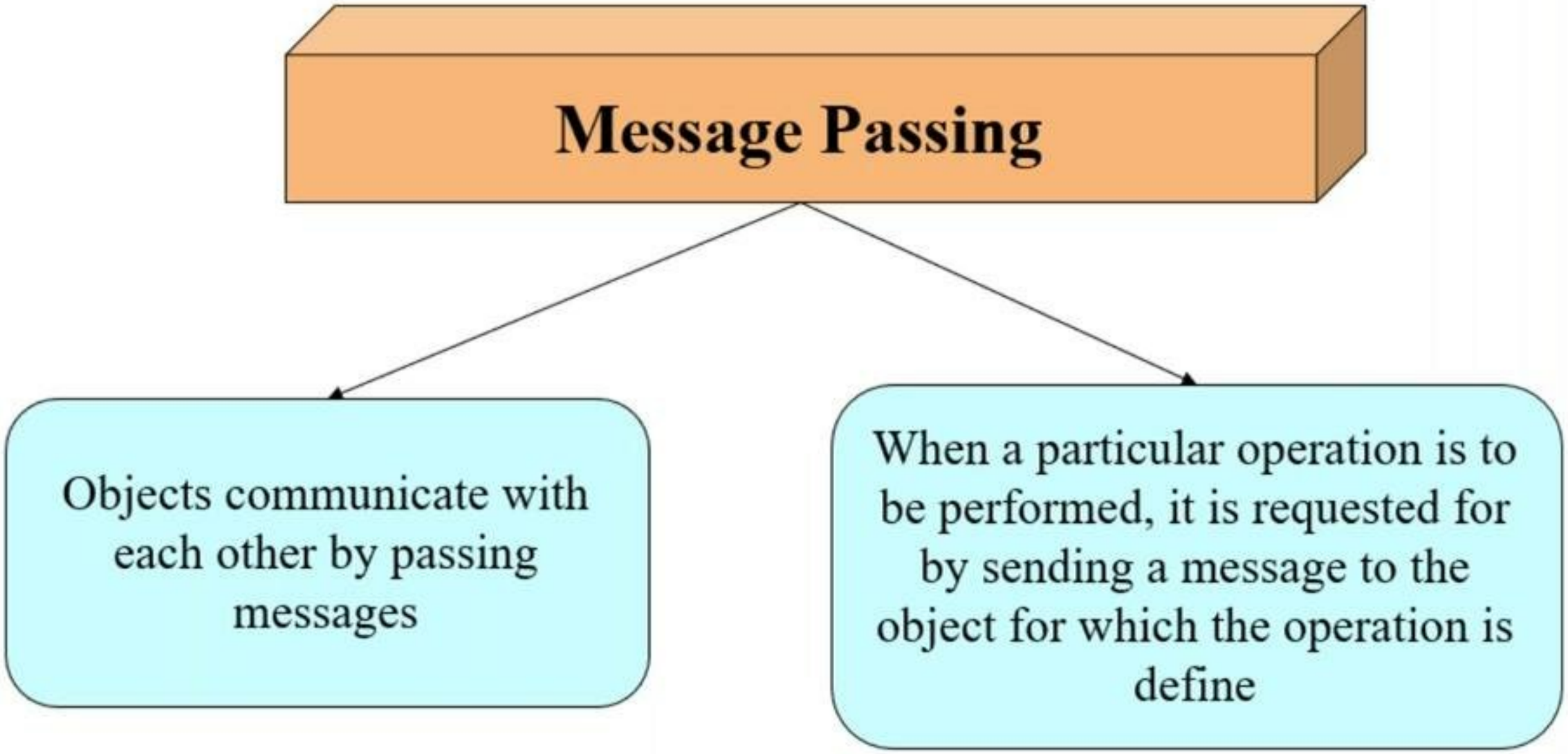
Object


Object
Core of Object oriented programming
Represents an entity in the real world
Provides practical basis for computer applications
Accomplishes specific tasks

Example of an Object



Message Passing



Difference between Class and Objects

Class and Objects

Class defines an entity

Object is the actual entity

Class is a conceptual model that defines all the characteristics and actions required of an object

All objects belonging to the same class have the same characteristics and actions

Implementing Classes in Java

Syntax

```
class <classname> {
        <body of the class>
}
```

where,

- class is the keyword used for creating a class,
- <classname> is the name of the class, and
- <body of the class> consists of declaration of attributes and methods.

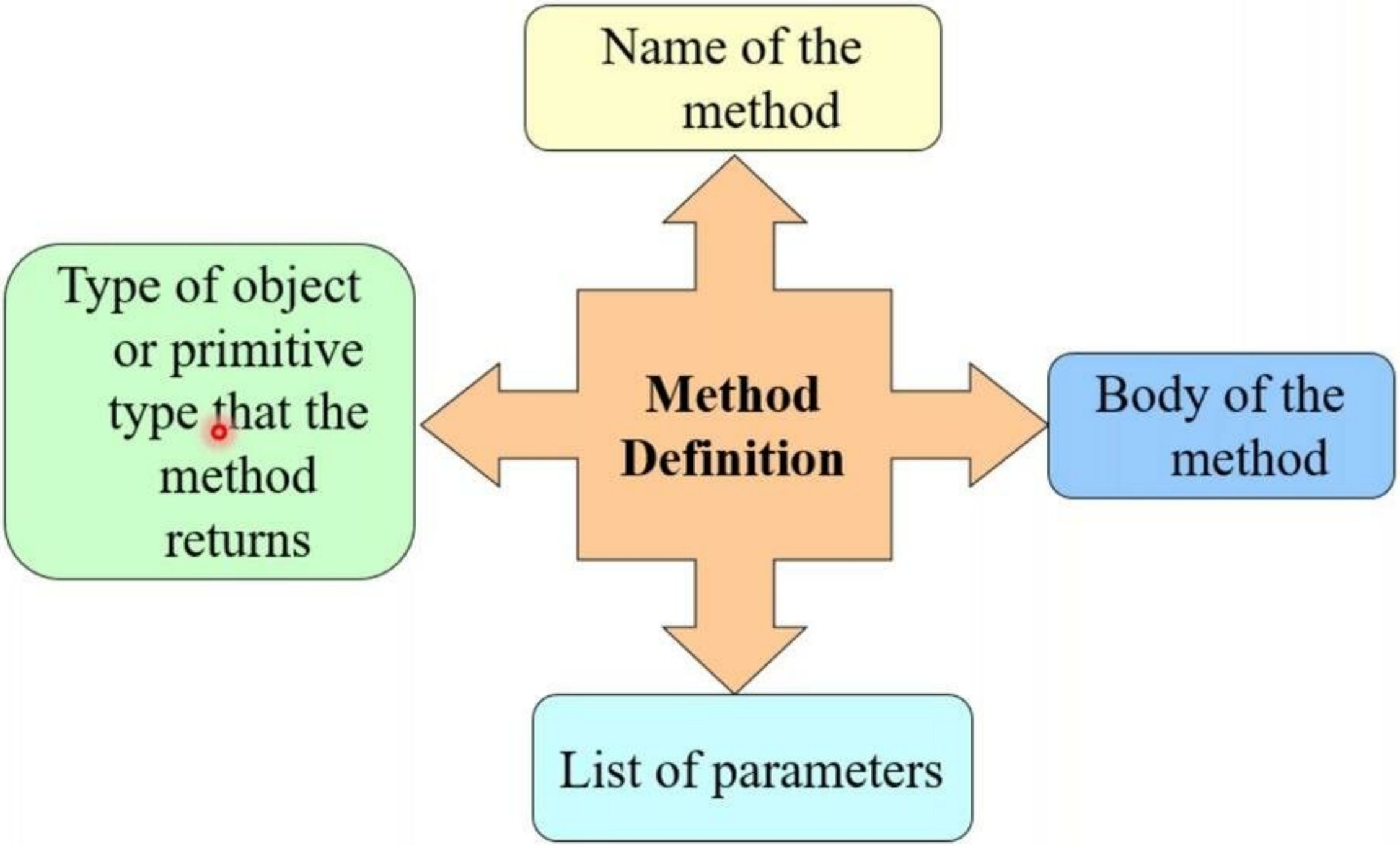
Implementing an Object

```
Class Employee
{
    <Body of the class>
    public static void main(String args[])
    {
        Employee obj=new Employee();
    }
}
```

New is a keyword

Employee is the name of the class

Methods in classes 5-1



Methods in classes 5-2

Syntax

<returntype> <methodname> (<type1> <arg1>, <type2> <arg3,…) {
<set of statements>
}

where,

returntype is the data type of the value returned by the method, <methodname> is the user-defined name of the method, and method's parameter list is a set of variable declarations.

Methods in Classes 5-3

```
class Employee {
  String empName;
  int empId;
   int salary;
   boolean available;
  void isAvailable() {
      if(available == true)
            System.out.println("The Employee is
  available");
  }
…
```

method

Methods in Classes 5-4

- Methods are accessed using dot notation.

- Object whose method is called is on the left of the dot, while the name of the method is on the right.

For Example,

Obj.isAvailable();

- Java provides class methods, which are similar to instance methods.

Methods in Classes 5-5

```
class Employee {
  String empId;
  String empName;
   int salary;
   boolean available;
}
```

PersonalDetails.java

Employee.java

Methods in Classes 5-5

```
class Employee {
  String empId;
  String empName;
   int salary;
   boolean available;
}
```

PersonalDetails.java

Employee.java

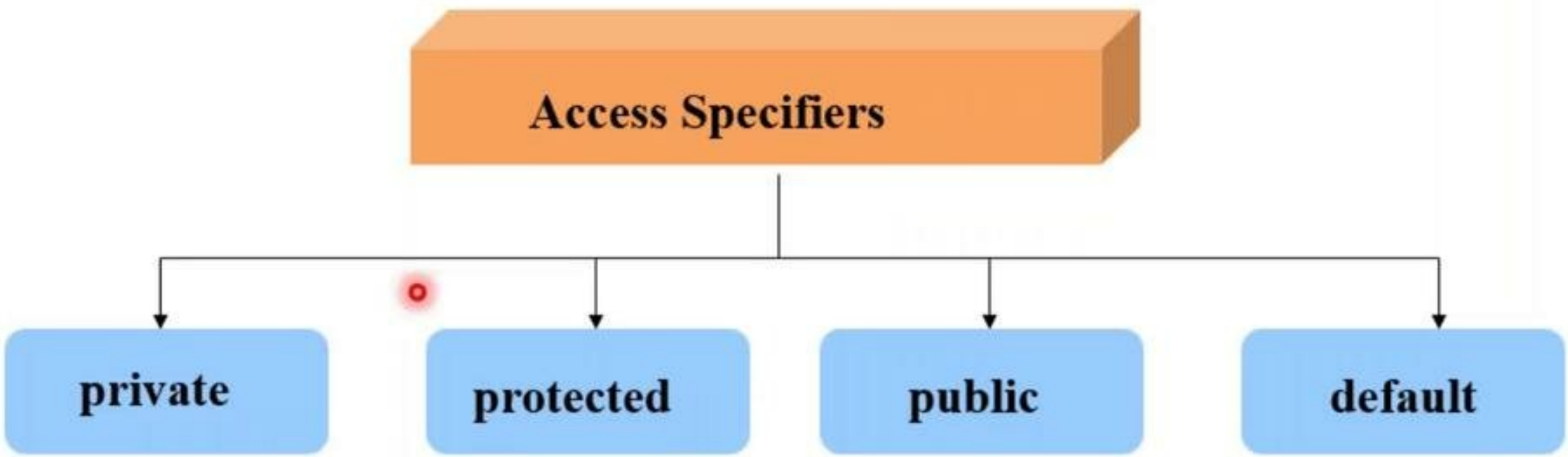
```
    boolean isAvailable() {
            if(available == true)
                    System.out.println("The Book is available");
return true;
    }
```

```
Employee objEmp = new Employee();
objEmp.isAvailable();
…..
```

Dot notation

Access Scope 2-1

- Information hiding is one of the most important features of OOPs.
- Reasons for information hiding are:
  - Changes made to any implementation details will not affect code that uses this class.
  - Prevents accidental erasure of data by users.
  - The class is easy to use.



Access Scope 2-2

public

Accessible to members and non members of the class

private

Accessible only to members of the class

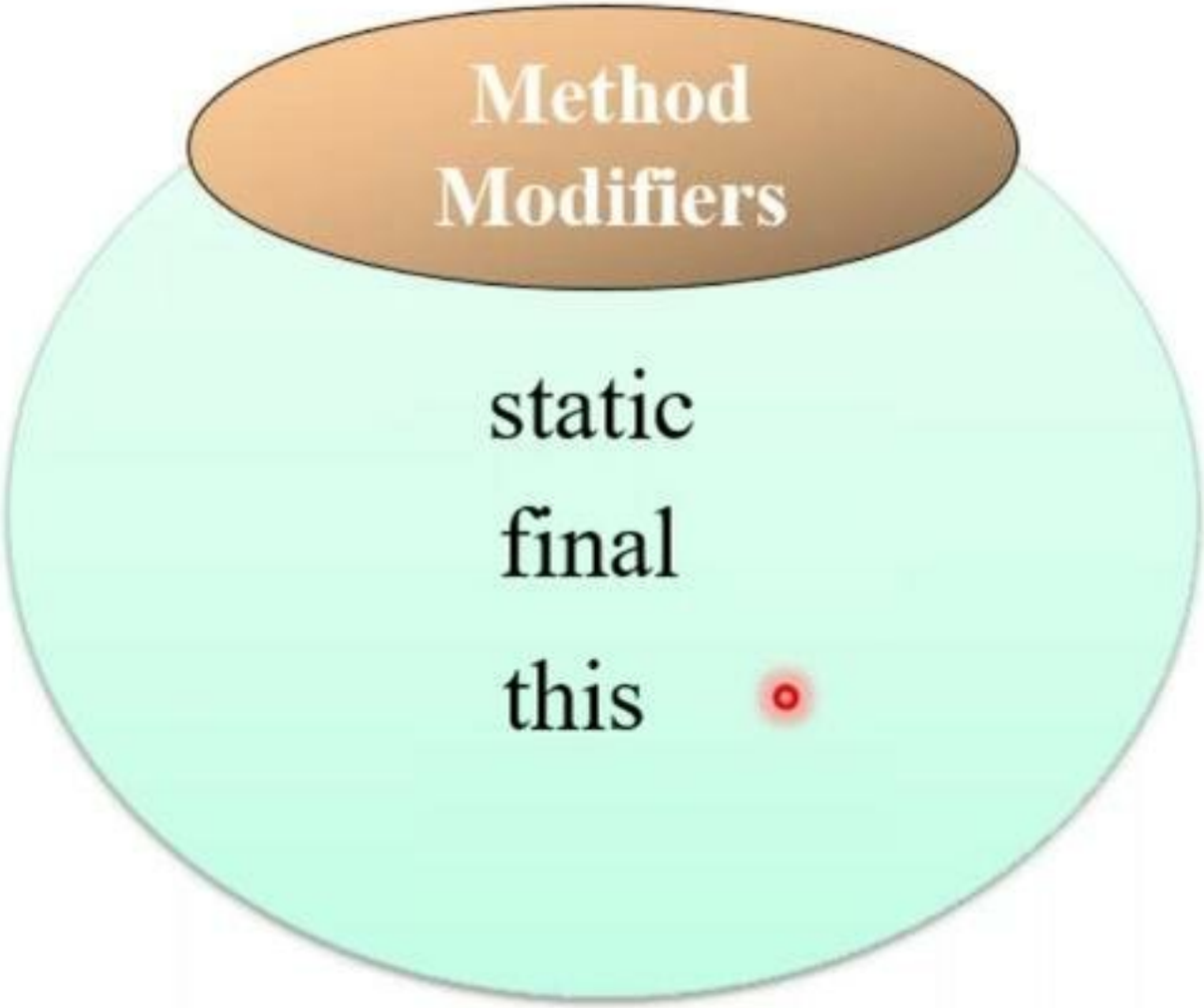
protected

Accessible to members of the class and members of its subclass

default

Accessible to members of the class in the same package

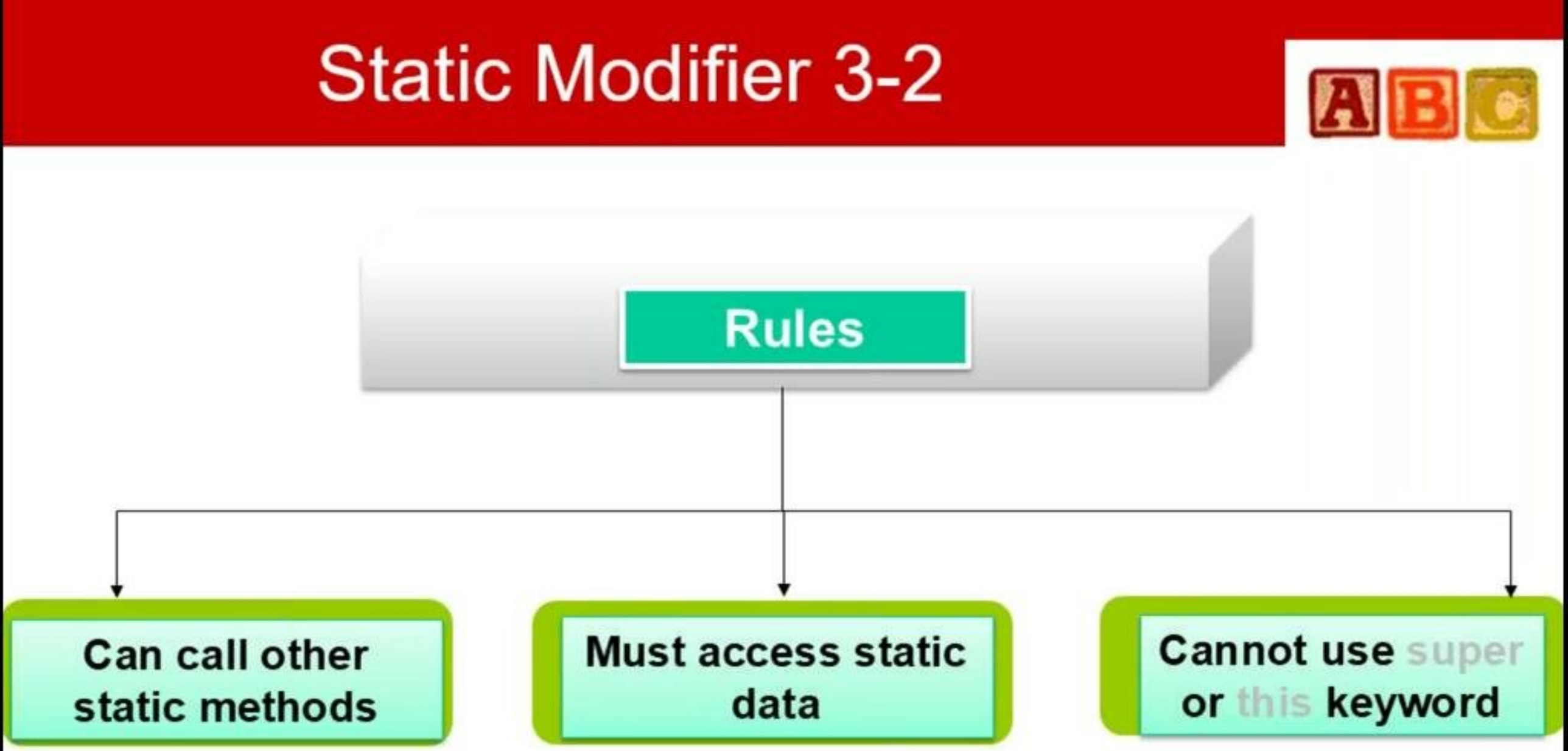
Method Modifiers



Static Modifier 3-1

- Programmer might need to access a class member independent of any object of the class.
- The keyword static is used to create such a member.
- Such a member is accessed before any object of the class is created.
  - For example, main() method is declared static as it can be invoked by the Java runtime system without creating an instance of the class.

Static Modifier 3-2



- Syntax for invoking a static method is:
  - `Classname.methodname();`

Declaring a static method

- Syntax for *declaring* a static method (writing down the recipe):

```
public class <class name> {

    public static void <method name> () {
        <statement>;
        <statement>;
        ...
        <statement>;
    }
}
```

- Example:

```
public static void companyInformation() {
    System.out.println("The Name of head HR is Mr Anupam Jauhari ");
}
```

Calling a static method

- Syntax for *calling* a static method (cooking using the recipe):
  - In another method such as main, write:

    ***<method name>* ();**

- Example:

  companyInformation();

- You can call the method multiple times.

  companyInformation();
  companyInformation();

Employee.java (Employee.java.JAV)

When to use static methods

When to use static methods

Place statements into a static method if:

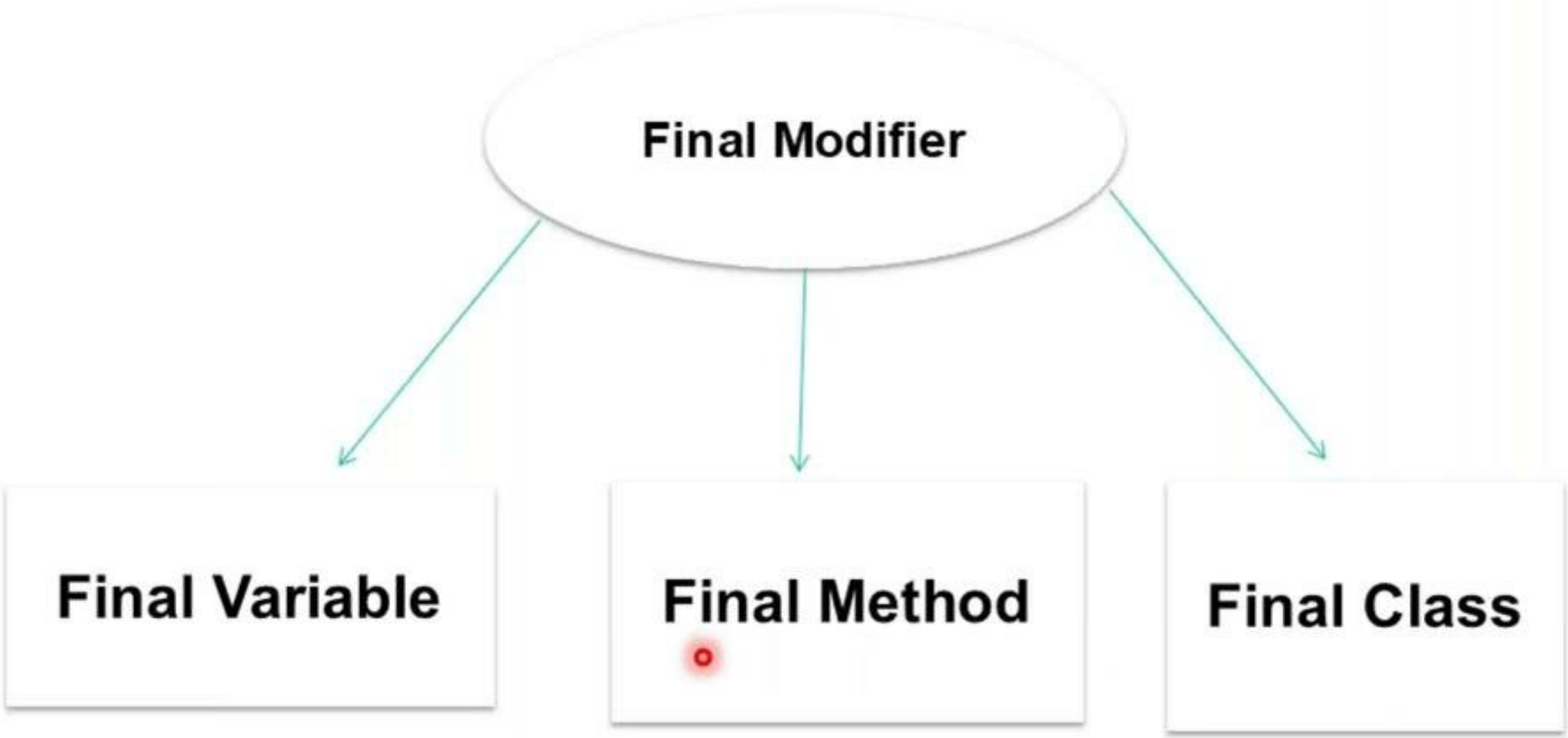
- The statements are related to each other and form a part of the program's structure, or
- The statements are repeated in the program.

You need not create static methods for:

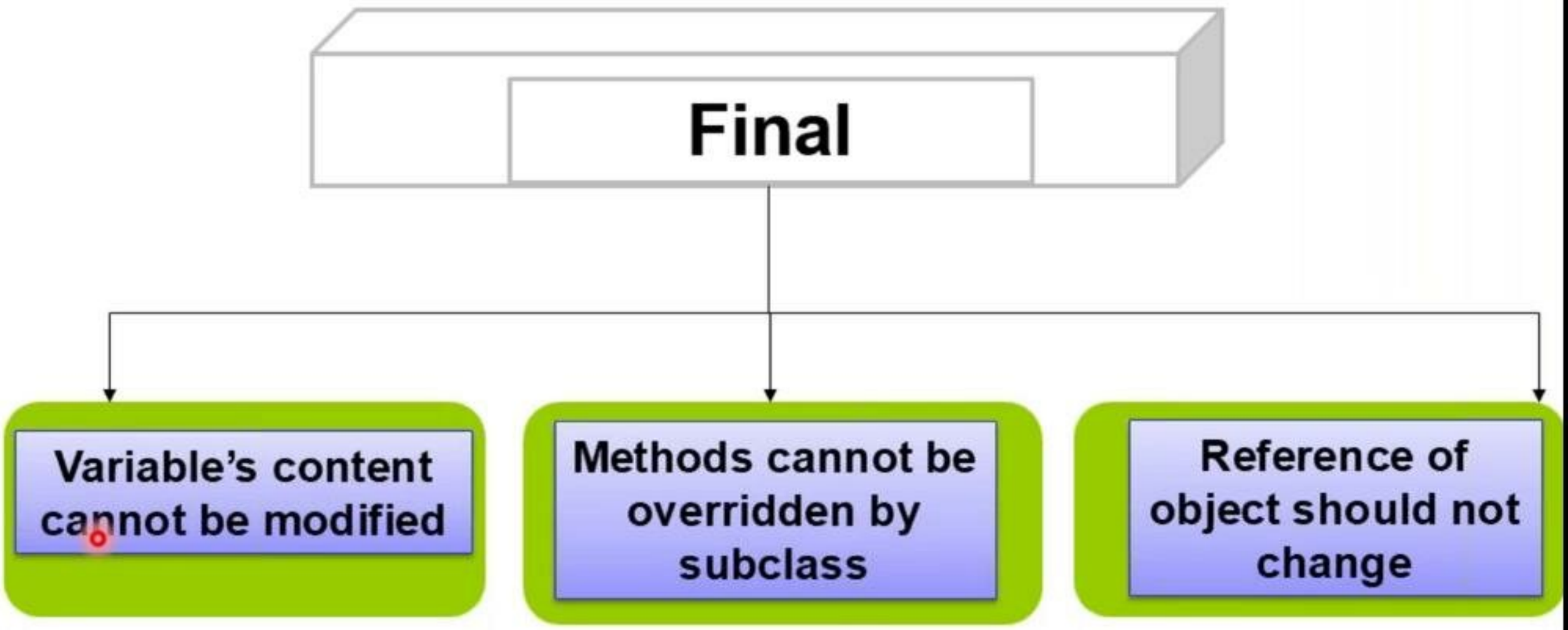
- Individual statements only occurring once in the program.
  (A single println in a method does not improve the program.)
- Unrelated or weakly related statements.
  (Consider splitting the method into two smaller methods.)
- Only blank lines.
  (Blank println statements can go in the main method.)

Final Modifier

It is used before a variable, method, and classes so that they cannot be changed latter.



Final Modifier

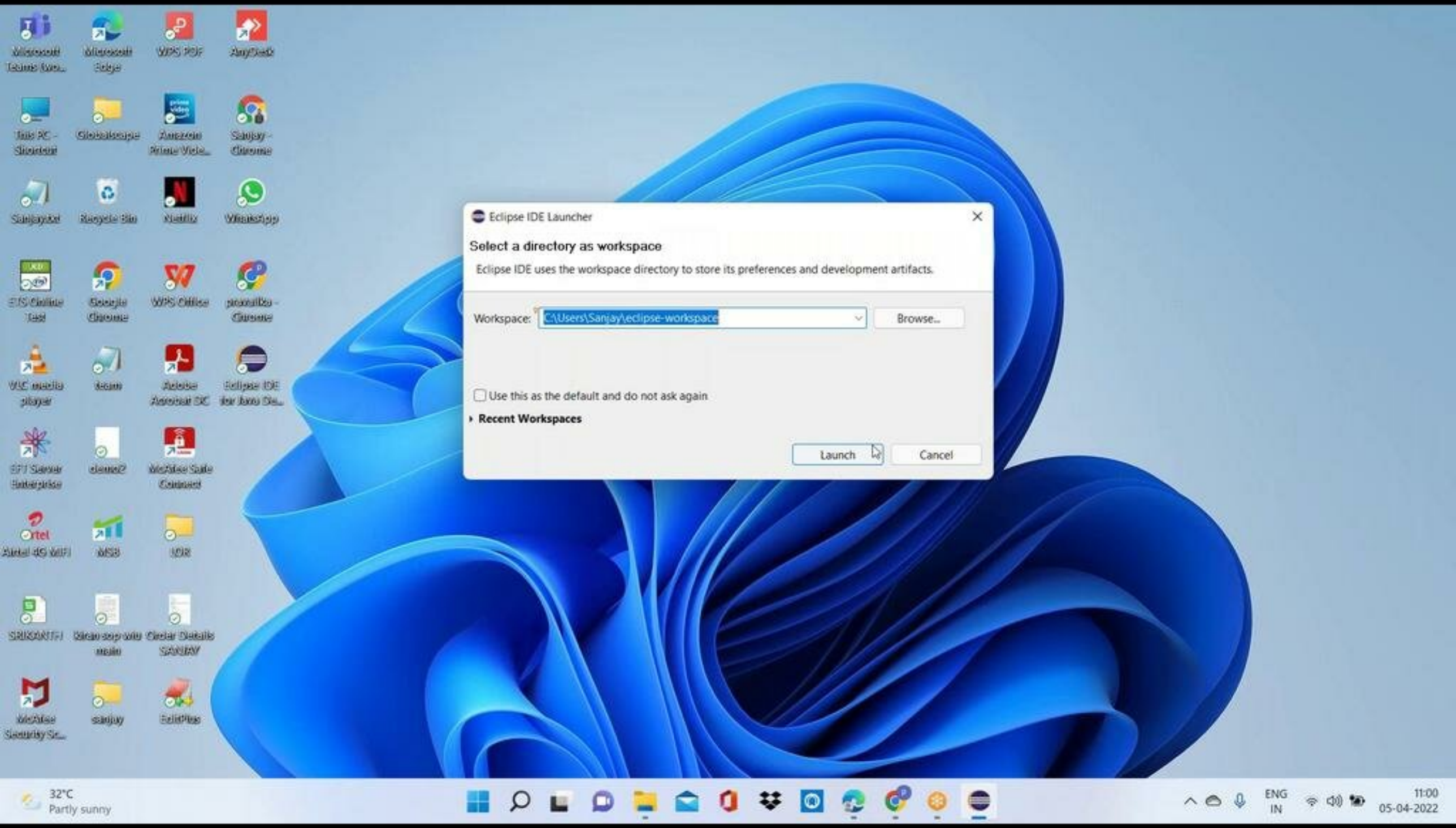


Example of this Keyword

```
class Employee {
    int age,eid;
    void init (int age, int eid)
    {
        this.age = age;
        this.eid = eid;
    }
}

public static void main (String args[])
   {
     Employee employee = new Employee();
     employee.init (30,2120703);
   }
```

Reference to an object


eclipseIDE
2022-03
32°C
Partly sunny
ENG
IN
11:00
05-04-2022


Eclipse IDE Launcher
Select a directory as workspace
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.
Workspace:
C:\Users\Sanjay\eclipse-workspace
Browse...
Use this as the default and do not ask again
Recent Workspaces
Launch
Cancel
32°C
Partly sunny
ENG
IN
11:00
05-04-2022


eclipseIDE
2022-03
Starting Eclipse IDE
32°C
Partly sunny
ENG
IN
11:00
05-04-2022

DXC - EmployeeManagementSystem/src/com/training/Person.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Ex...

- 00001.Greetings
- EmployeeManagementSy
  - JRE System Library [jre
  - src
    - com.training
      - Person.java

*Person.java

```
28
29      //Copy constructor
30      public Person(Person person) {
31          this.firstName = person.firstName;
32          this.lastName = person.lastName;
33          this.age = person.age;
34          this.contactNumber = person.contactNumber;
35      }
36
37      @Override
38      public String toString() {
39          return "\nfirstName=" + firstName + ", lastName=" + lastName + ", age=" + age + ", contactNumber="
40                  + contactNumber;
41      }
42      public static void main(String[] args) {
43          Person person1=new Person();
44          System.out.println(person1);
45          Person person2=new Person("Sita","Rani",34,1223344556l);
46          System.out.println(person2);
```

Problems Javadoc Declaration Console

<terminated> Person [Java Application] D:\eclipse-jee-2021-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (05-Apr-2022, 11:29:39 am – 11:29

```
firstName=Sanjay, lastName=Kumar, age=43, contactNumber=9818254421

firstName=Sita, lastName=Rani, age=34, contactNumber=1223344556

firstName=Sanjay, lastName=Kumar, age=43, contactNumber=9818254421
```

Writable | Smart Insert | 44 : 37 [1231]