CPE207 Object Oriented Programming

Week 5

Static keyword, Nested Classes, Enum types, Deeper in classes

Dr. Nehad Ramaha,
Computer Engineering Department
Karabük Universities

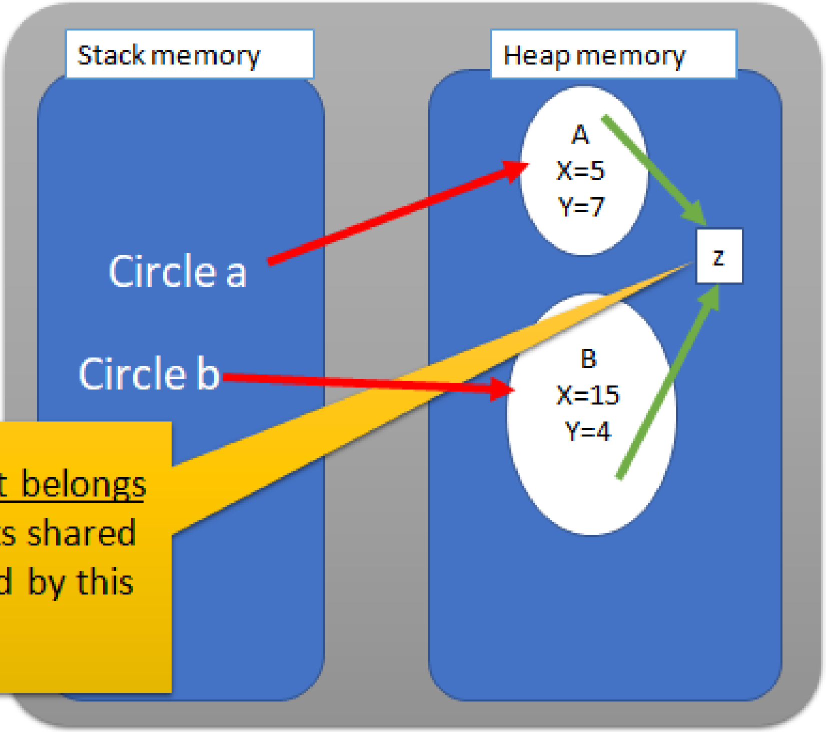
These Slides mainly adopted from Assist. Prof. Dr. Ozacar Kasim lecture notes

The class notes are a compilation and edition from many sources. The instructor does not claim intellectual property or ownership of the lecture notes.

Static keyword

- The **static keyword** is a non-access modifiers and used in java mainly for memory management. It is used with variables, methods, blocks and nested classes.
- It is a keyword that **are used for share the same variable or method of a given class.**
- **This is used for a constant variable or a method that is the same for every instance of a class.**

```
class Circle{

private int x;
private int y;
static int z;

}
```



Here z is static variable: it belongs to class, not an object. Its shared by all the objects created by this class.

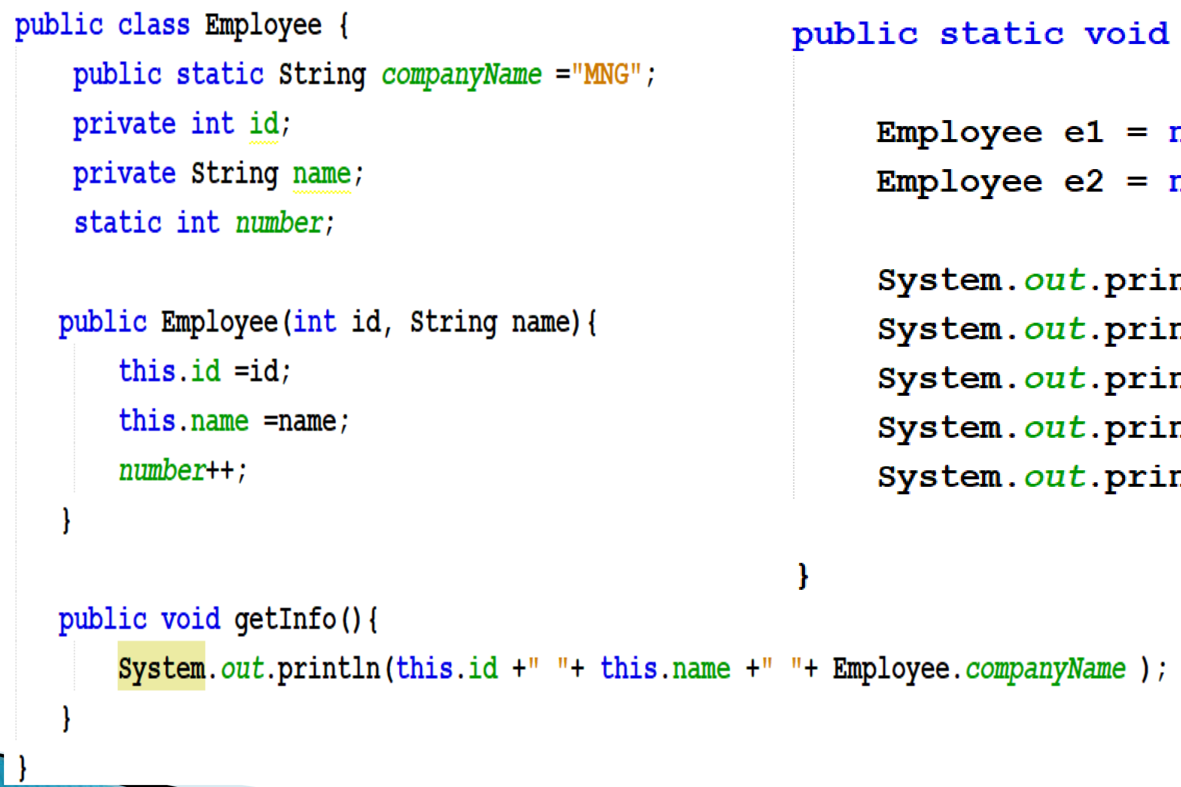
Static variable

- If any variable, we declared as static is known as static variable.
- Static variable is used for fulfill the common requirement. For Example, college name of students etc.
  Name of the college is common for all students.
- **The static variable allocate memory only once in class area at the time of class loading.**
- Using static variable we make our program memory efficient (i.e it saves memory).

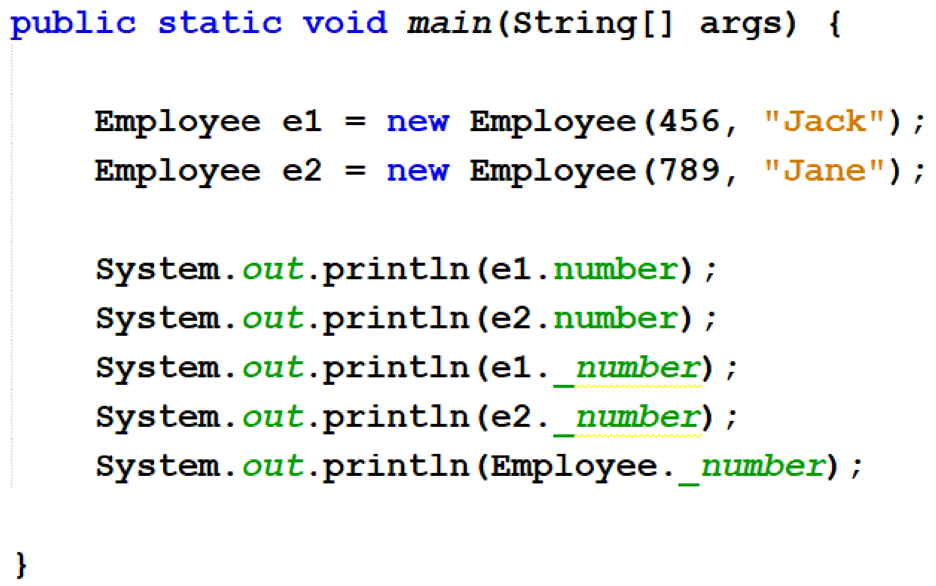
When and why we use static variable?

- Suppose we want to store record of all employee of any company, in this case employee id is unique for every employee but company name is common for all. <u>So, use static variable to store the company name</u>.
- When we create a static variable as a company name then only once memory is allocated otherwise it allocate a memory space each time for every employee object.

An example of static keyword



```
public class Employee {
    public static String companyName ="MNG";
    private int id;
    private String name;
    static int number;

   public Employee(int id, String name){
       this.id =id;
       this.name =name;
       number++;
   }

   public void getInfo(){
       System.out.println(this.id +" "+ this.name +" "+ Employee.companyName );
   }
}
```



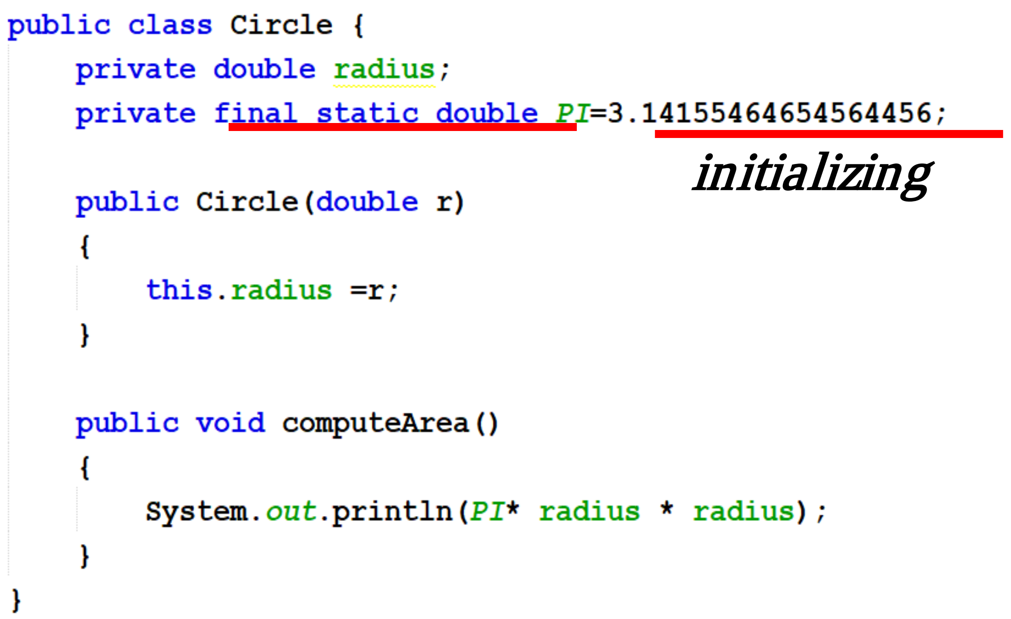
```
public static void main(String[] args) {

    Employee e1 = new Employee(456, "Jack");
    Employee e2 = new Employee(789, "Jane");

    System.out.println(e1.number);
    System.out.println(e2.number);
    System.out.println(e1._number);
    System.out.println(e2._number);
    System.out.println(Employee._number);

}
```

final Keyword

- **static** keyword always fixed the memory that means **that *will be located only once in the program***
- **final** keyword always fixes the value that means **it makes variable values constant**
- Note: As for as real time statement there concern every final variable should be declared the static but there is no obligation that every static variable declared as final.

Java Constants : final Keyword for variables

- When a variable is declared with *final* keyword, its value can’t be modified, essentially, a constant.
- This also means that you must initialize a final variable.
  - *If you cannot change it then you must initialize it, right!*



```
public class Circle {
    private double radius;
    private final static double PI=3.1415546465456456;

    public Circle(double r)
    {
        this.radius =r;
    }

    public void computeArea()
    {
        System.out.println(PI* radius * radius);
    }
}
```

What is Static Method in Java?

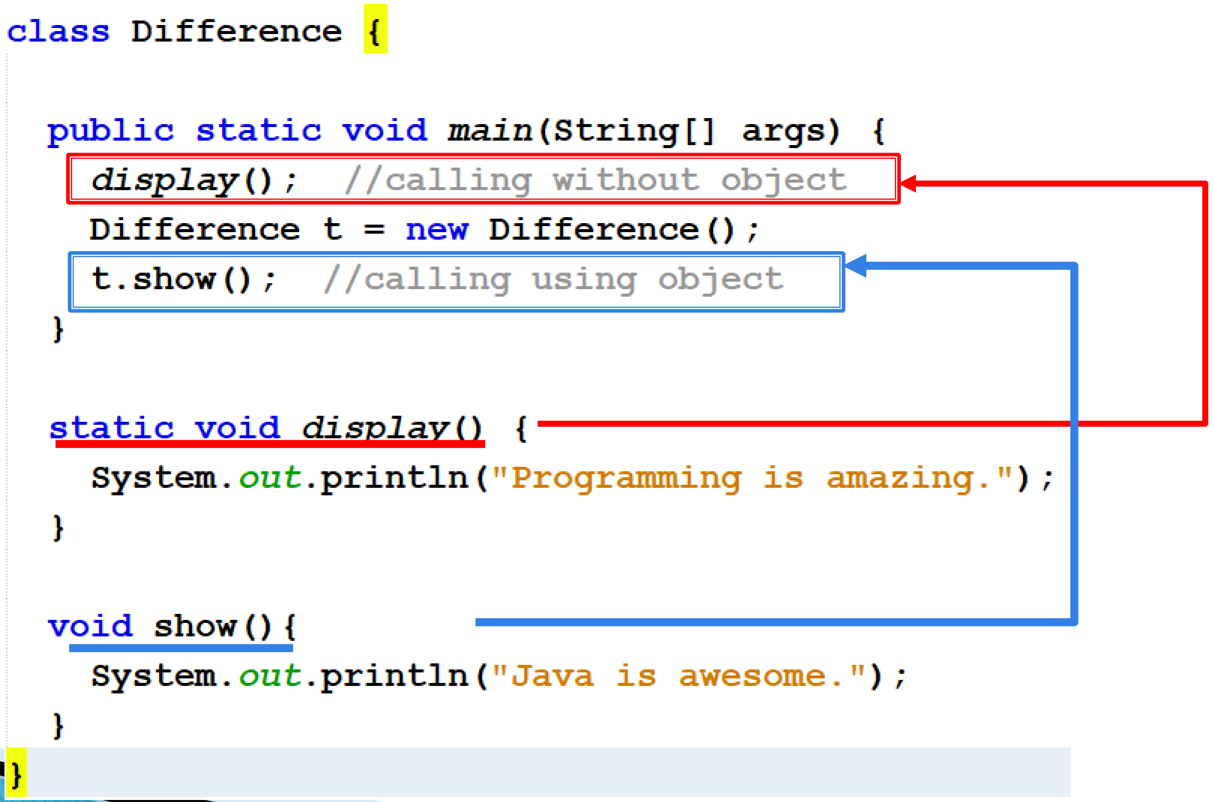
- Static method is a method which belongs to the class and not to the object(instance)
- A static method can access only static data. It can not access non-static data (instance variables)
- A static method can call only other static methods and can not call a non-static method from it.
- A static method can be accessed directly by the class name and doesn’t need any object
- A static method cannot refer to "this“ keyword.

	Non-Static method	Static method

1	These method never be preceded by static keyword
	Example:
	`void  fun1()`
	`{`
	`......`
	`......`
	`}`
		These method always preceded by static keyword
		Example:
		`static  void   fun2()`
		`{`
		`......`
		`......`
		`}`

2	Memory is allocated multiple time whenever method is calling.	Memory is allocated only once at the time of class loading.
3	It is specific to an object so that these are also known as instance method.	These are common to every object so that it is also known as member method or class method.
4	These methods always access with object reference Syntax: **Objref.methodname();**	These property always access with class reference Syntax: **className.methodname();**
5	If any method wants to be execute multiple time that can be declare as non static.	If any method wants to be execute only once in the program that can be declare as static .

Static Method vs Non-Static (Instance) Method



```java
class Difference {

  public static void main(String[] args) {
    display();  //calling without object
    Difference t = new Difference();
    t.show();  //calling using object
  }

  static void display() {
    System.out.println("Programming is amazing.");
  }

  void show(){
    System.out.println("Java is awesome.");
  }
}
```

Another Example

```
class MyClass{
    private static int data;

    public static void setData(int d) {
        data = d;
    }

    public static int getData() {
        return data;
    }
}
```

```
public class JavaApp {


    public static void main(String[] args) {
        MyClass.setData(50);
        System.out.println(MyClass.getData());
    }
}
```

No need to create an object

We know the static keyword now.

So, why is the main method static?

Nested Classes

- The Java allows you to define a class within another class. Such a class is called a *nested class*
- Nested classes are divided into two categories:
  - non-static. Non-static nested classes are called inner classes.
  - Static: Static nested classes are called static nested classes

```
class OuterClass {
    ...
static class
StaticNestedClass {
        ...
    }
}
```

```
class OuterClass {
    ...
    class InnerClass {
        ...
    }
}
```

Inner class

- As with instance methods and variables, an inner class is associated with an instance of its enclosing class(outer) and has direct access to that object's methods and variables.
- Also, because an inner class is associated with an instance, it cannot define any static members itself.

```
class OuterClass {
    ...
    class InnerClass {
        ...
    }
}
```

Inner classes

```
    OuterClass outerObject = new OuterClass(); //you need to create an object first
    OuterClass.InnerClass innerObject = outerObject.new InnerClass();
```

Static Nested Classes

- As with class methods and variables, a static nested class is associated with its outer class.
- And like static class methods, a static nested class cannot refer directly to instance variables or methods defined in its enclosing class: it can use them only through an object reference.

```
class OuterClass {
    ...
static class
StaticNestedClass{
        ...
 }
}
```

```
//you do not need to create and object from OuterClass
OuterClass.StaticNestedClass nestedObject = new OuterClass.StaticNestedClass();
```

Why Use Nested Classes?

•If a class is useful to only one other class, then it is logical to embed it in that class and keep the two together.

•**It increases encapsulation**: (inner class can be private)

•**It can lead to more readable and maintainable code**:

https://docs.oracle.com/javase/tutorial/java/javaOO/nested.html

Inner Class

```
class OuterClass {
  int x = 10;

class InnerClass {  //what if this is private?
    int y = 5;
  }
}

public class MyMainClass {
  public static void main(String[] args) {
    OuterClass myOuter = new OuterClass();
    OuterClass.InnerClass myInner = myOuter.new InnerClass();
    System.out.println(myInner.y + myOuter.x);
  }
}
```

Static nested Class

```
class OuterClass {
  int x = 10;

  static class InnerClass {
    int y = 5;
  }
}

public class MyMainClass {
  public static void main(String[] args) {
    OuterClass.InnerClass myInner = new OuterClass.InnerClass();
    System.out.println(myInner.y);
  }
}
```

Enum Types

- An *enum type* is a special data type that enables for a variable to be **<u>a set of predefined constants.</u>**
- The variable must be equal to one of the values that have been predefined for it.
- Common examples include compass directions (values of NORTH, SOUTH, EAST, and WEST), the days of the week and so on.

Syntax

```
enum Day {
SUNDAY,
MONDAY,
TUESDAY,
WEDNESDAY,
THURSDAY,
FRIDAY,
SATURDAY
}//simple enum
```

Enum Types with Constructor

A Java enum type can have a private constructor that can be used to initialize instance variables(attributes).

```
public enum Branch {
    MATH(001),
    PHYSICS(002),
    GEOMETY(003)
   ; // semicolon needed when fields / methods follow
 private int fieldId;
    Branch(int fieldId){

        this.fieldId =fieldId;
    }
}
```

values() method can be used to return all **values** present inside **enum**

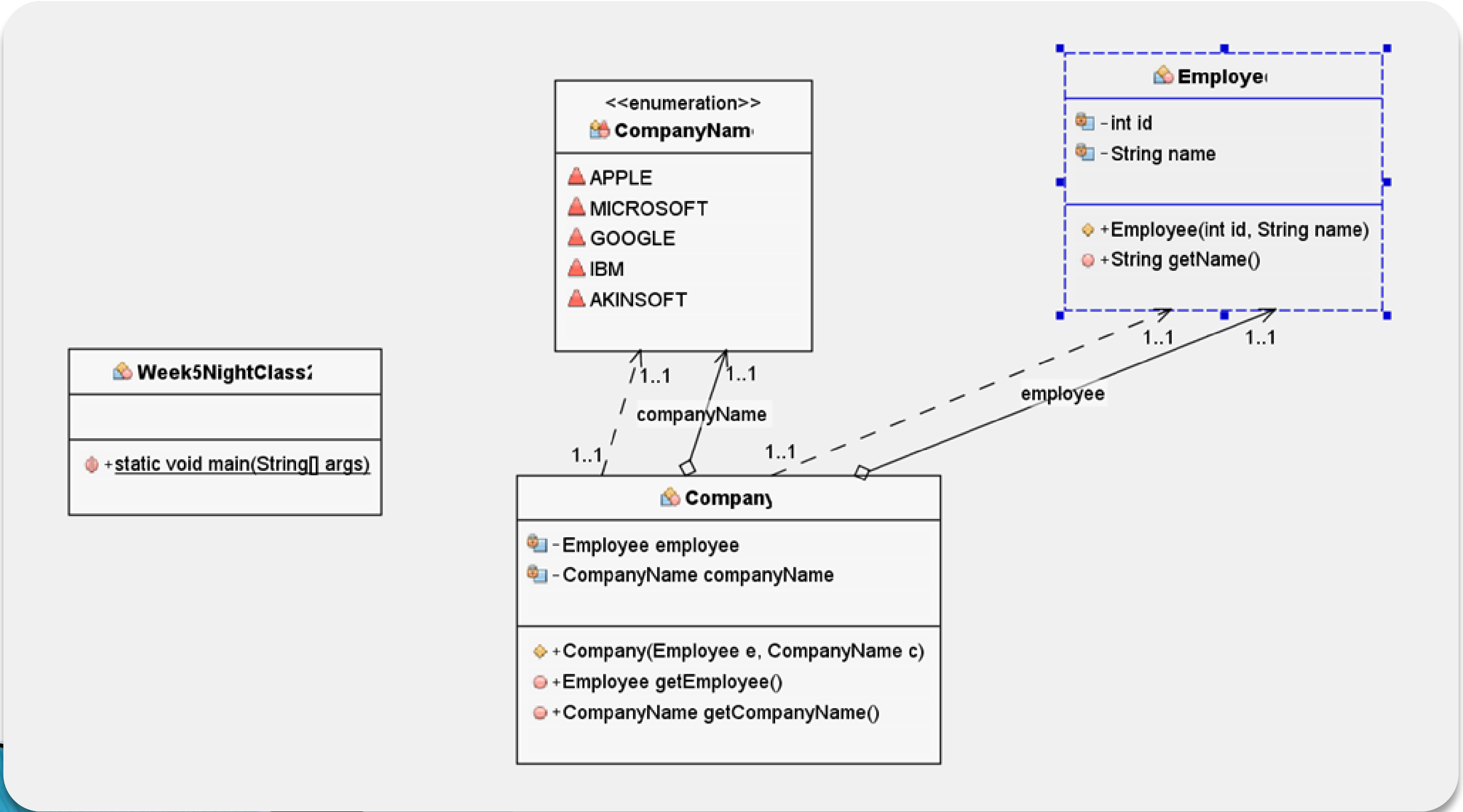
Looping through enum items

```
Branch[] brunches= Branch.values() ;

for(Branch b : brunches) {
    System.out.println(b);
}
```

Enum types with Classes: Making CompanyName

```
enum CompanyName{
GOOGLE(1995, "Google was founded in 1998 by Larry Page and Sergey Brin while"),
MICROSOFT(1975, "Microsoft Corporation is a technology company with headquarters in
Redmond, Washington");

private int createdYear;

final private String description;

private CompanyName(int cYear, String desc) {
    this.description = desc;
    this.createdYear = cYear;
    }

    public String getDescription(){
        return this.description;
    }

}
```

Example: Lets move to NetBeans!



Thanks ☺

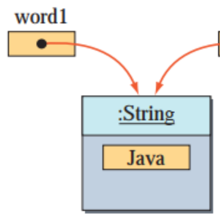
Lab Exercise-1: static keyword

- Create a Circle class where
  - declare a private constant double variable PI has value of 3,141519
  - declare a private variable called radius.
  - class constructor will have an argument to set radius.
  - declare a method called computeArea() to compute area of a circle object.
  - create 3 different circle instances, with radiuses 5, 10, 15.
  - print all the areas using a foreach loop.

Lab Exercise-2: Enum types

1. Create Branch Enum type will contain following branches
   1. MATH(“information regarding math”)
   2. PHYSICS(“information regarding physics”)
   3. CS(“information regarding cs”)
   4. ENG(“information regarding eng”)
2. Create a Teacher Class which contains id (int), and branch (Enum) attributes. The class must have a constructor with these two parameters.
3. Create four teacher objects; each has different branch.
4. Put them all in an array, and print their branches using for loop.

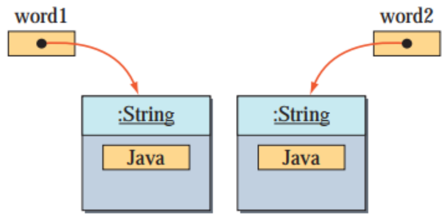
Case A: Referring to the same object.



```
word1 == word2            is true
word1.equals( word2 )     is true
```

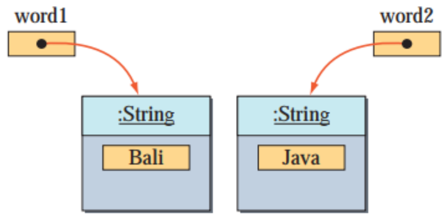
Note: If **x == y** is **true**, then **x.equals(y)** is also **true**. The reverse is not always true.

Case B: Referring to different objects having identical string values.



```
word1 == word2            is false
word1.equals( word2 )     is true
```

Case C: Referring to different objects having different string values.



```
word1 == word2            is false
word1.equals( word2 )     is false
```