UNTAR
Universitas Tarumanagara

Terakreditasi BAN PT A (Unggul) QS STARS RATING SYSTEM 2019 ★★★ JABEE CPA AUSTRALIA ICAEW CHARTERED ACCOUNTANTS

UNTAR untuk INDONESIA

Object-based Programming

Week 5 – Static, Abstract, and Interfaces

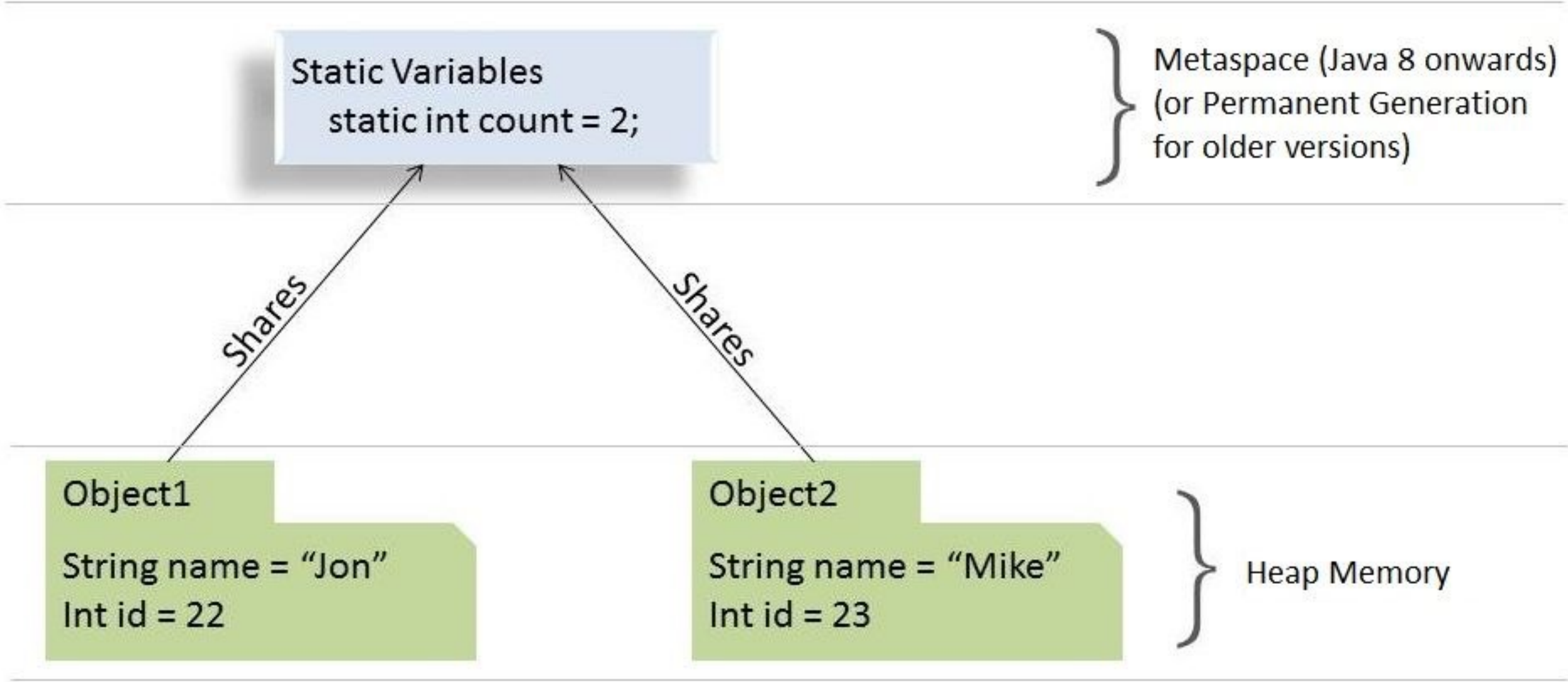
Presentation (random group)

- Static and non-static variables
- Static and non-static methods
- Abstract class & methods
- Interface

static vs non-static variables (1)

```
public class Item {
    public String name;
    public static int numInstances = 100;
}

public class Main {
    public static void main(String[] args) {
        System.out.println(Item.name);          // error
        System.out.println(Item.numInstances);  // 100
    }
}
```


Static Variables
static int count = 2;
Metaspace (Java 8 onwards)
(or Permanent Generation
for older versions)
Shares
Shares
Object1
String name = “Jon”
Int id = 22
Object2
String name = “Mike”
Int id = 23
Heap Memory

static vs non-static variables (2)

```
public class Item {
    public String name;
    public static int numInstances = 100;
}
```

```
public class Main {
    public static void main(String[] args) {
        Item item01 = new Item();
        Item item02 = new Item();

        item01.name = “Indomie”;
        item01.numInstances = 42;

        System.out.println(item02.name);          // “”
        System.out.println(item02.numInstances); // 42
    }
}
```

static vs non-static methods

```
public class Item {
    public void display() {
        ...
    }

    public static void print() {
        ...
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Item.display();              // error
        Item.print();                // ok
    }
}
```

Abstract method

- A signature definition of a method, but with no implementation

Abstract class

- A class that contains one or more abstract method definitions

```
public abstract class Item {
    protected int value;
    protected String creator;

    public abstract void displayDetails();

    public int getValueInIDR() {
        return value * 14500;
    }
}
```

```
public abstract class Item {
    protected int value;
    protected String creator;

    public abstract void displayDetails();

    public int getValueInIDR() { ... }
}
```

```
public class Vase extends Item {

    // must implement displayDetails() or error
    public void displayDetails() {
        System.out.println("This vase was created by " + creator);
        System.out.println("It has a value of " + value +
                           " US dollars");
    }

}
```

Interfaces

- Multiple inheritance
- Specification of some functionality that a class is intended to provide
- Only specification

```
public interface MyInterface {
    // interface attribute
    public String defaultMessage = “Hello World”;

    // interface method
    public void displayMessage();
}
```

```
public interface MyInterface {
    // interface attribute
    public String defaultMessage = “Hello World”;

    // interface method
    public void displayMessage();
}
```

```
public class Message implements MyInterface {
    private String content;
    private String from;
    private String to;

    public void displayMessage() { // must implement this!
        System.out.println(content);
    }
}
```

```
public interface MyInterface { ... }
public interface AnotherInterface { ... }

public class Message implements MyInterface, AnotherInterface {
    private String content;
    private String from;
    private String to;

    public void displayMessage() { // must implement this!
        System.out.println(content);
    }
}
```

```
public class AClass { ... }
public interface MyInterface { ... }
public interface AnotherInterface { ... }


public class Message extends AClass implements MyInterface, AnotherInterface {
    private String content;
    private String from;
    private String to;

    public void displayMessage() { // must implement this!
        System.out.println(content);
    }
}
```