**Administrative**

**Today’s session**

Exam 1 key

Mutator and accessor methods

Object creation and deletion

Constructor methods

Memory regions

Homework 4

**Session Topics**

**Exam 1 key**

● The Exam 1 key is NOT available on Blackboard.

**Mutator and accessor methods**

**Mutator method**

● A **mutator method** provides a way for the instance variables of one object to be changed by other objects within an application.

● A mutator method is also known as a **setter method**.

● A mutator method is nearly always declared with the **public** modifier.

● There is typically one mutator method for each instance variable.

● *setter method* example:

public void setCount(int count)

{

this.count = count;

}

● The **this** keyword enables the Java Virtual Machine to distinguish between an instance variable and a local variable with the same name.

● this. refers to the instance variable.

**Accessor method**

● An **accessor method** provides a way for the instance variables of one object to be viewed by other objects within an application.

● An accessor method is also known as a **getter method**.

● An accessor method is nearly always declared with the **public** modifier.

● There is typically one accessor method for each instance variable.

● *getter method* example:

public int getCount()

{

return count;

}

● See **Classes – one file** sample application on Blackboard.

● See **Classes – two files** sample application on Blackboard.

**Object creation and deletion**

**Object creation**

● The **new** operator creates an object from a class:

<Class-name> <object-variable> = new <Class-name>();

OR

<Class-name> <object-variable> = null;

…

<object-variable> = new <Class-name>();

● Once an object is created:

✓ It may be referred to by its name.

✓ Its instance variables may be referred to with:

<object-name>.<instance-variable>

✓ Its methods may be called with:

<object-name>.<method-name>(<parameter-list>);

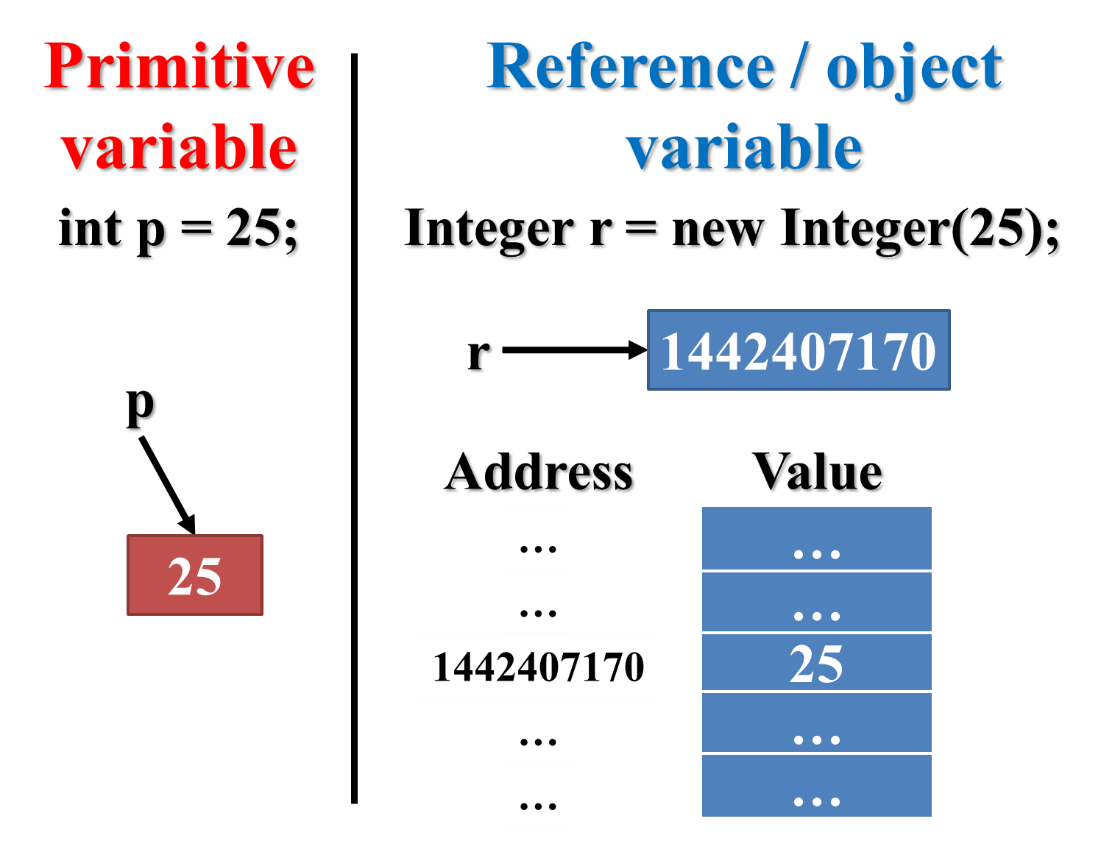
● When a primitive type variable is created (declared), it points to the single spot where its value is stored. A variable reference returns the stored value.

● When a reference type variable is created, it points to a memory address. A variable reference returns the memory address.

● In this graphic:

✓ p is a **primitive variable** that points directly to a value.

✓ r is a **reference / object variable** that points to a memory address.



**Object deletion**

● Assigning the **null** value to an object deletes it:

<object-variable> = null;

● A **null value** is a special value that may be assigned to an object.

● A null value clears the memory address of an object.

● An object doesn’t have to be explicitly deleted since, when an application ends, all objects are automatically deleted.

● See **Classes – one file** sample application on Blackboard.

**Constructor methods**

● A **constructor method** specifies how an object is initialized.

● A constructor method has the same name as the class but no return type.

● If a class does not specify a constructor method, Java creates one by default that initializes all instance variables.

● A constructor method is often overloaded so that an object may be created in multiple ways.

● Constructor methods example:

public class MyClass

{

// Declare instance variables

private int ivVar1;

private int ivVar2;

// Constructor method, no parameters

public MyClass()

{

ivVar1 = -1;

ivVar2 = -1;

}

// Constructor method, one parameter

public MyClass(int ivVar1)

{

this.ivVar1 = ivVar1;

ivVar2 = -1;

}

// Constructor method, two parameters

public MyClass(int ivVar1, int ivVar2)

{

this.ivVar1 = ivVar1;

this.ivVar2 = ivVar2;

}

…

}

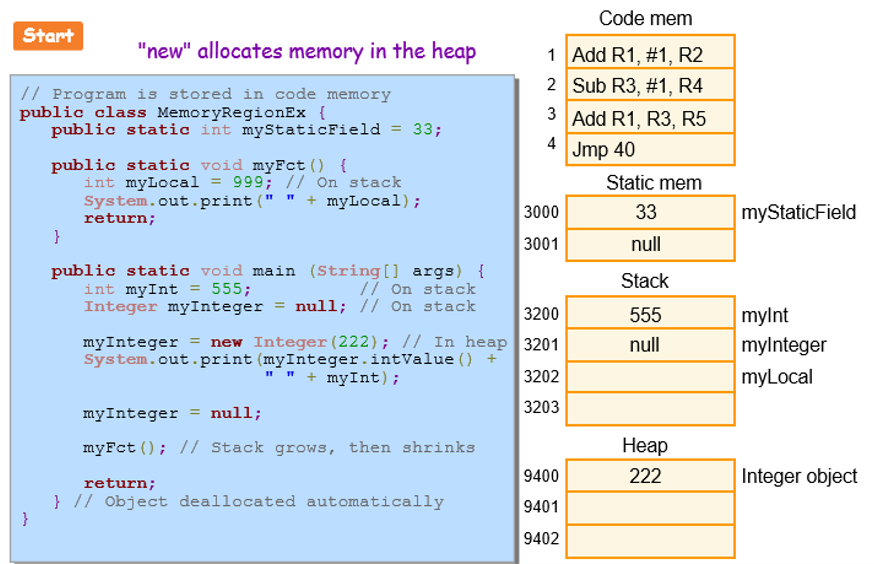
● See **Classes – two files** sample application on Blackboard.

**Memory regions**

● The Java Virtual Machine (JVM) maintains four memory regions for each running application:

|  |  |
| --- | --- |
| Modifier | Purpose |
| Code | To store program instructions. |
| Static memory | To store static fields and static local variables. |
| Stack | To store method calls, including (non-static) local variables. |
| Heap | To store objects created with the new operator. |

● JVM memory example:



● See **Heap watch** sample application on Blackboard.

**Homework 4**

● Assigned today.

● Available on Blackboard.

● Due in two weeks.