hi-tech college + JOHN BRYCE Leading in IT Education a matrix company

Identifiers, Keywords and Types

Comments

- There are three types of comments in java:
  - /* C style comments */
  - // C++ style comments
  - /** Java documentation comment
    used for the Javadoc tool */
- The Javadoc tool is used for generating class documentation.

A Statement

- The statement syntax in Java is identical to the statement syntax in C or C++.
- Every sentence (one or more lines of code) must terminate with a semicolon (;).
- White spaces may be entered between statement's parts.
- Example:
  - *result = num1 + num2 + num3 – num4 ;*

Blocks

- Blocks syntax is identical to blocks syntax in C or C++.
- Every new scope (e.g., the beginning of a method) must starts with an opening brace ({) and ends with a closing one (}).
- White spaces may be used inside blocks.
- Blocks can be nested.

Blocks – an Example

```
public class Car {
      private int max_speed;
     private String model;
          . . .
     public float get_fuel_consumption( ) {
           . . .
      }
}
```

Identifiers

- Are names that we give to variables, classes, or methods.
- Can start with a Unicode letter, underscore (_) or a dollar sign ($).
- Are case-sensitive and have no maximum length.
- Examples:
  - UserName
  - userName
  - user_name
  - _sys_var1
  - $change

Java Keywords

abstract	double	interface	package	synchronized
boolean	default	if	private	this
byte	do	implements	protected	throw
break	extends	import	public	throws
char	else	instanceof	return	transient
case	false	int	short	try
catch	final	long	static	true
class	finally	null	strictfp	void
continue	float	native	super	volatile
	for	new	switch	while

Primitive Types

- *boolean* – This is a logical type that may hold either “true” or “false”.
- *char* – A textual type that may hold a 16-bit unicode character.
  - e.g. char ch = ‘a’ ;

Primitives - Integral Types

- *byte* – A 8 bit integer, ranges: -2^7 to 2^7-1
- *short* – A 16 bit integer, ranges: -2^{15} to $2^{15}-1$
- *int* – A 32 bit integer, ranges: -2^{31} to $2^{31}-1$
- *long* – A 64 bit integer, ranges: -2^{63} to $2^{63}-1$

Primitives: Floating Types

- *float* – A 32 bit floating point value.
- *double* - A 64 bit floating point value.

Integral Types – cont'd

- A leading 0 indicates an octal value.
- A leading 0x indicates a hexadecimal value.
- Integrals have a default type of int.
- All integrals types in java are signed numbers.

- Example:

```
int decVal = 26; // The number 26, in decimal
int octVal = 032; // The number 26, in octal
int hexVal = 0x1a; // The number 26, in hexadecimal
```

Integral Types – Cont'd

- A suffix l or L represents a long int type.

  E.g., *long l_num = 5L ;*
- A suffix s or S represents a short int type.

  E.g., *short s_num = 4s ;*

- Example:

```
short s_num = 7.5 ;  //An error, can not
                     //assign an int value
                     //into a short int type.
```

Floating Types -Suffix and Default Types.

- A default floating point number is of type *double*.
- A suffix f or F represents a *float* type.

  E.g. float f_num = 5.4f ;

- A suffix d or D represents a *double* type.

  E.g. double d_num = 4.8d ; // d is optional since double is default.

float f_num = 7.5 ; *// An error, can not*
// assign a double value
// into a float type.

References to Objects

- Objects are created dynamically on the heap.
- References (which may be local or global) are used as handles to objects.
- A reference holds an object address which is used without pointers notation.

References Vs. Primitives

```
public class Example {
    private int counter=0;                                  //counter is a
    primitive.
    private Box b1;                                         //b1 is a null reference
                                                            //to a Box object.

    public Example(int height, int width, int length)   {
      b1= new Box(height,width,length);                      //b1 is now
                                                             //initialized and points
                                                            //to the Box object
                                                            //located on the heap.

  }
  public static void main(String args[ ]) {
    Example e1=new Example(3,6,5);                          //e1 is a reference.
    …
  }
   …
}
```

Instantiation Of an Object

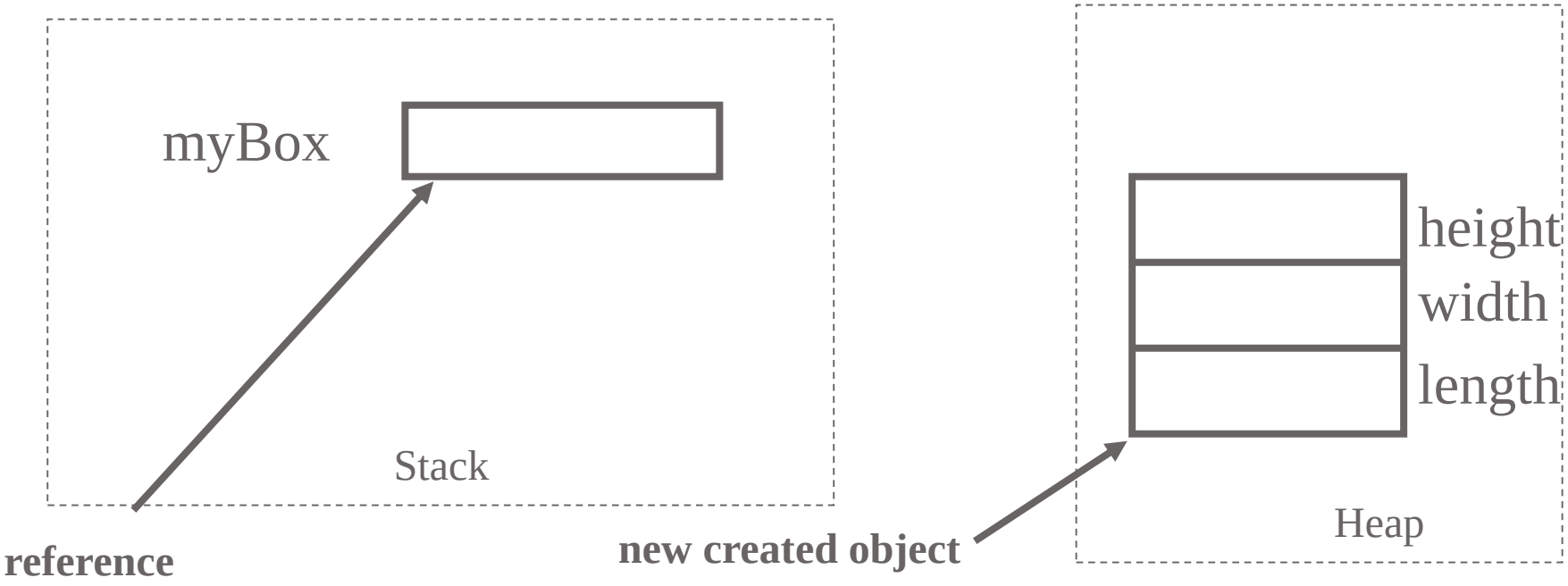
- Instantiation of an object is done using the operator *new .*

E.g: *new MyClass(firstArg, SecArg);*

- This will result in the following sequence:
  - Memory is allocated for the new object on the heap.
  - Data members are initialized to their default values.
  - A constructor is executed.
  - The object is assigned to the reference that from now on will be used as its handle.

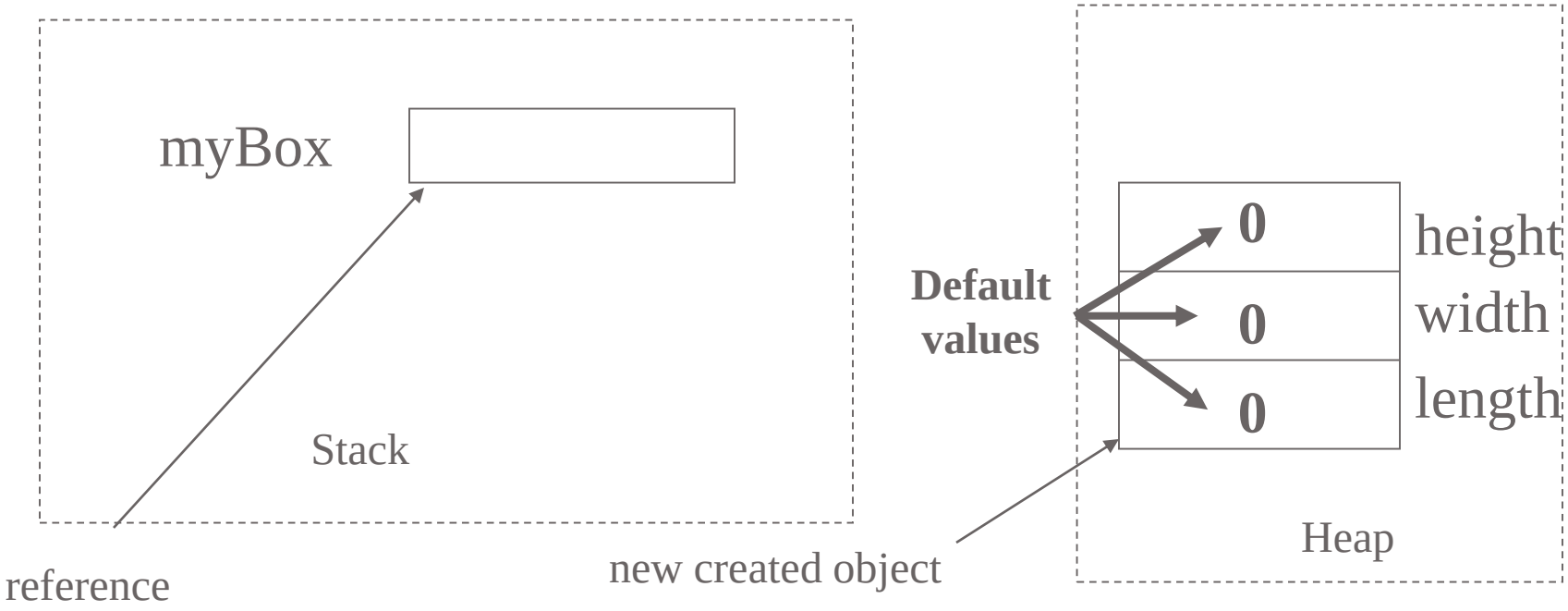
Step 1 – Memory allocation

Box myBox=new Box(5,6,7);



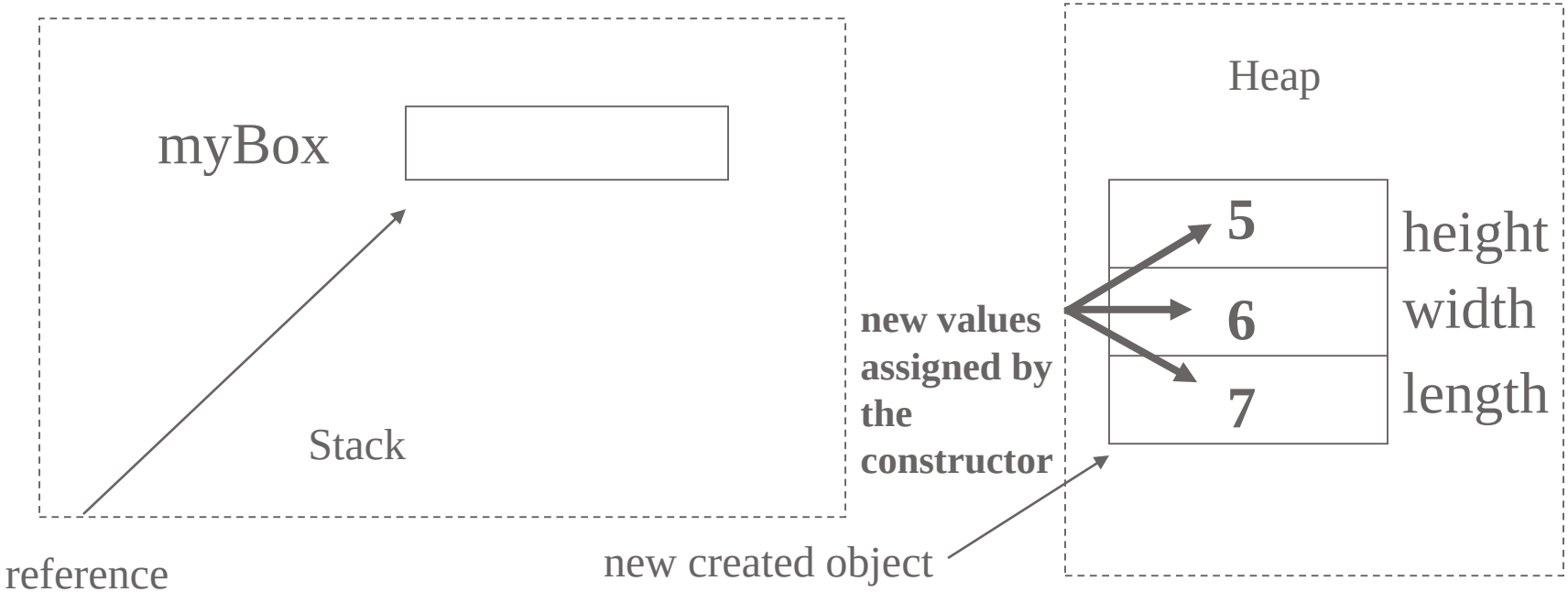
Step 2 – Data members are initialized to their default values

Box myBox=new Box(5,6,7);



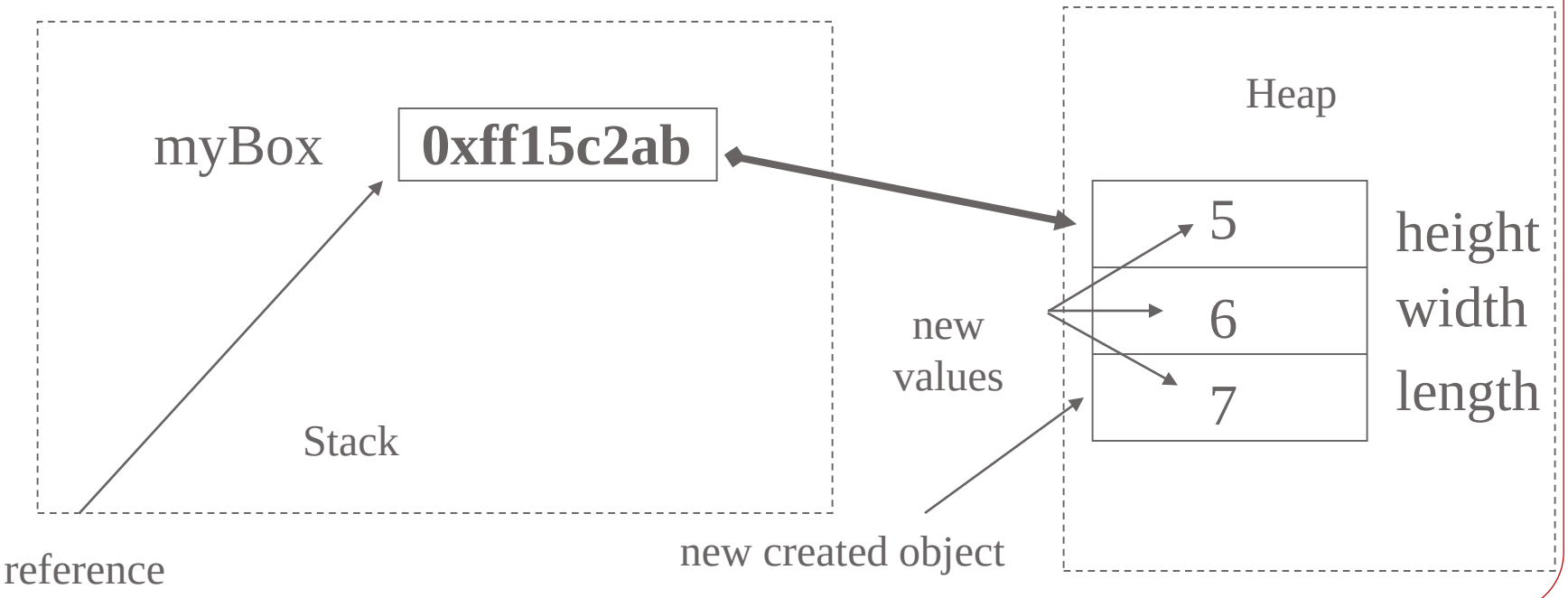
Step 3 – Constructor is executed

Box myBox=new Box(5,6,7);



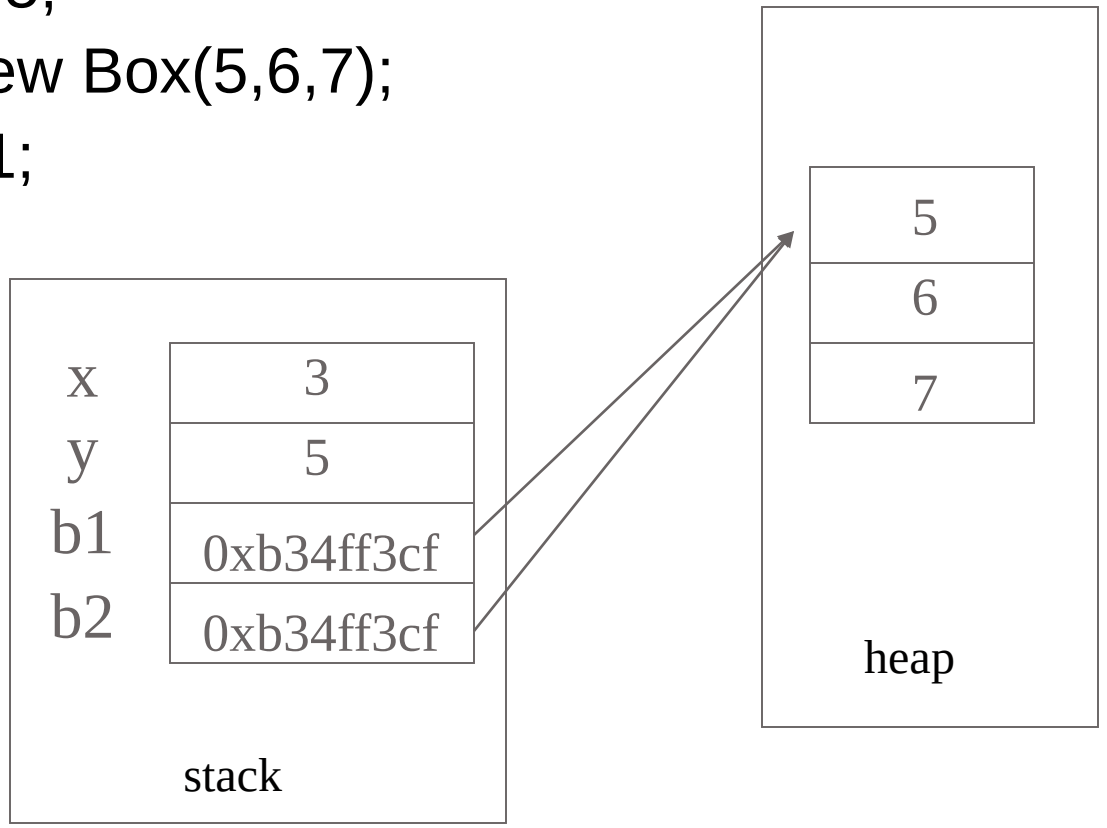
Step 4 – The newly created object is assigned to its reference

Box myBox=new Box(5,6,7);



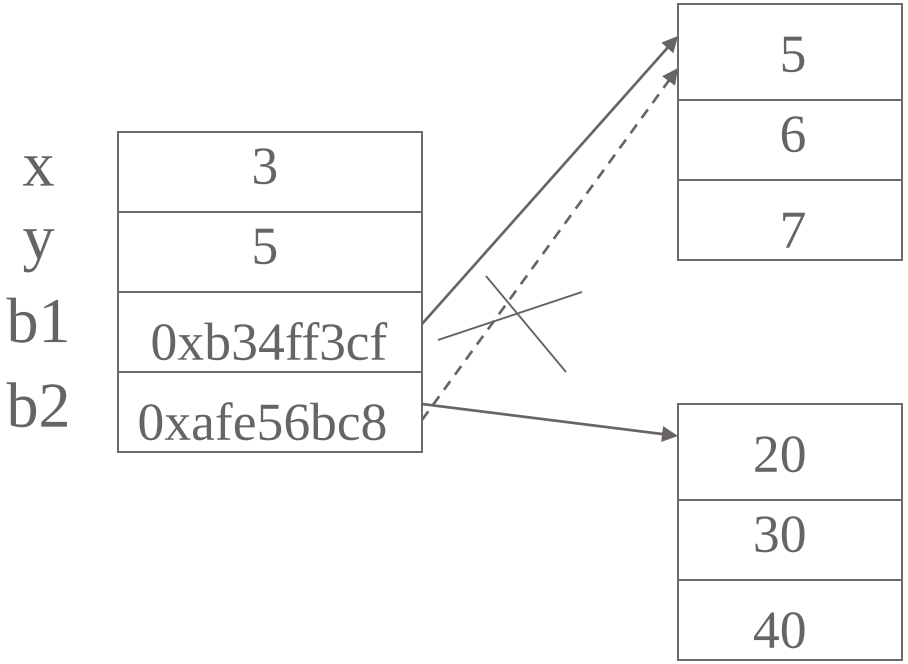
References in Action

```
int x=3, y=5;
Box b1=new Box(5,6,7);
Box b2=b1;
```



Reassignment Of a Reference

```
int x=3, y=5;
Box b1=new Box(5,6,7);
Box b2=b1;
   …
b2=new Box(20,30,40);
```



Methods Arguments

- The only way to pass arguments to methods in Java (except for a distributed environment) is **call by value**.
- When a primitive is sent to a method as an argument, a copy of it is created and passed by value to the method.
- When a reference is sent to a method as an argument, a copy of the reference is created and passed by value to the method.
- The method's argument is another reference to the original object. Both references may affect the same object.
- Objects can not be passed to methods (only their references can).

Call By Value - Example

```
public class CallByValue {
     public static void changeVal(int num){
        num=3;
    }
     public static void changeRef(Box b1){
        b1=new Box(2,2,2);
    }
     public static void changeObjectAttributes(Box b1)     {
         b1.setHeight(9);
     }
    static public void main(String args[])    {
        int num=5;
        Box b1=new Box(1,1,1);                              //height=1,width=1,length=1;
        changeVal(num);
        System.out.println(num);                            //What will be printed?
        changeRef(b1);
        System.out.println(b1.getArea());                   //What will be printed?
        changeObjectAttributes(b1);
        System.out.println(b1.getHeight());                 //What will be printed?
       }
}
```

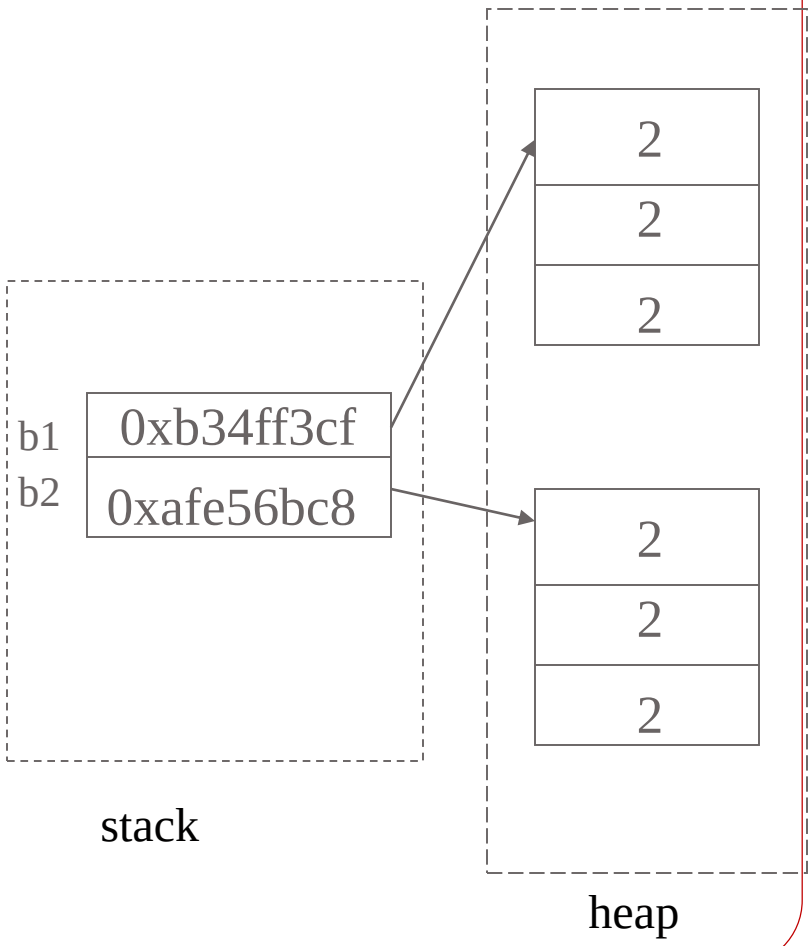
The *this* Reference

- A reference to the current object (the object that invoked the method).
- Accepted by every non static method as the first argument.
- May be used as a reference to the current object when invoking another method or another constructor.
- May be used in order to distinguish between a data member and a local variable with the same name.

*Note: static methods do not have a **this** reference. static methods will be explained later.*

this In Action

```
public class Box {
    private int height;
    private int width;
    private int length;
    public Box(int height,int width,int length) {
        this.height=height;
        this.width=width;
        this.length=length;
    }
     public Box(Box bArg) {
        this.height=bArg.height;
        this.width=bArg.width;
        this.length=bArg.length;
    }
    public Box replicateBox(){
        Box tmpBox=new Box(this);
        return tmpBox;
    }
  public static void main(String args[ ]) {
      Box b1=new Box(2,2,2);
      Box b2=b1.replicateBox();
      //. . . rest of main.
  }
}
```



The *this* Reference-cont'd

Original code

```
public class ThisExample {
    int num;
    public void chgNum(int num)
    {
        this.num=num;
    }
    static public void main(String args[])
   {
        ThisExample te=new ThisExample();
        te.chgNum(12);
   }
}
```

Code converted by the compiler

```
public class ThisExample {
    int num;
    public void chgNum(ThisExample this, int num)
    {
        this.num=num;
    }
    static public void main(String args[])
     {
        ThisExample te=new ThisExample( );
        chgNum(te, 12);
     }
}
```

Coding Conventions

- Packages:

  package shapes.colorShapes;

- Classes:

  class Box

- Interfaces:

  interface Fly

- Methods:

  getArea( )

Coding Conventions – cont'd

- Variables:

  rectHeight

- Constants:

  MAX_STUDENTS_IN_CLASS

  MIN_VAL