

HBO Graduaat Informatica Optie Programmeren

Java Basics

How to create a class



c v o l e e r s t a d

v o l w a s s e n e n o n d e r w i j s



Contents

- Class Construction
 - Class Declaration
 - Class Body
- Understanding Instance and Class Members
 - Instance variables & instance methods
 - Class variable & class methods
 - Objects vs. Classes





Goals

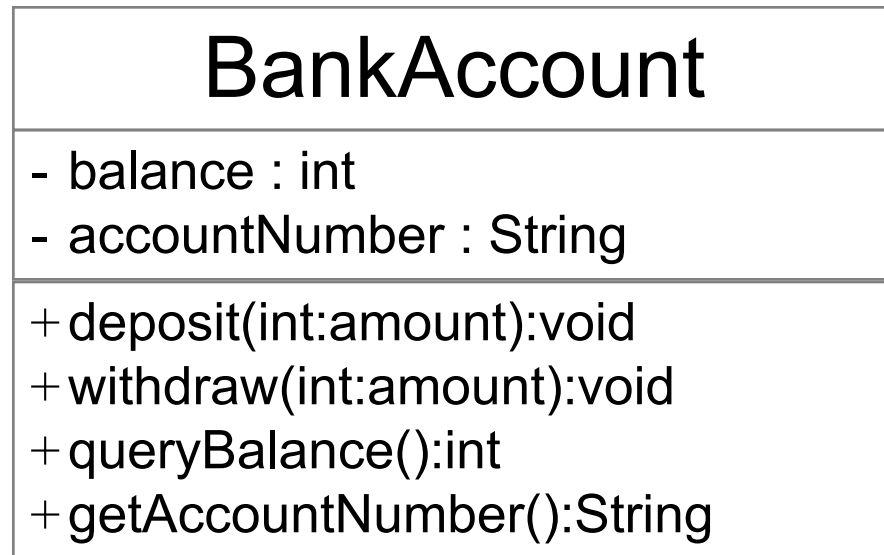
- Class Construction
 - To be able create a class and describe all element of a class
- Understanding Instance & Class Members
 - to be able distinguish Instance and Class Members
 - to be able distinguish Objects and Classes





Class Construction

- UML Representation





Class Construction

```
public class BankAccount
{
    private int balance;
    private String accountNumber;

    public BankAccount (String accountNumber)
    {
        this.accountNumber = accountNumber;
    }
    public String getAccountNumber()
    {
        return this.accountNumber;
    }
    public int queryBalance()
    {
        return this.balance;
    }
}
```

Bank Account

- balance : int
- accountNumber : String
- + deposit(int:amount):void
- + withdraw(int:amount):void
- + queryBalance():int
- + getAccountNumber():String

```
public void deposit (int amount)
{
    this.balance += amount;
}
public void withdraw (int amount)
{
    this.balance -= amount;
}
}
```



Class Construction

```
public class BankAccount
```

Class Declaration

```
{  
    private int balance;  
    private String accountNumber;  
  
    public BankAccount (String accountNumber)  
    {  
        this.accountNumber = accountNumber;  
    }  
    public String getAccountNumber()  
    {  
        return this.accountNumber;  
    }  
    public int queryBalance()  
    {  
        return this.balance;  
    }  
}
```

Bank Account

- balance : int
- accountNumber : String
- + deposit(int:amount):void
- + withdraw(int:amount):void
- + queryBalance():int
- + getAccountNumber():String

```
public void deposit (int amount)  
{  
    this.balance += amount;  
}  
public void withdraw (int amount)  
{  
    this.balance -= amount;  
}  
}
```



Class Construction

```
public class BankAccount  
{
```

Class Body

```
    private int balance;  
    private String accountNumber;  
  
    public BankAccount (String accountNumber)  
    {  
        this.accountNumber = accountNumber;  
    }  
    public String getAccountNumber()  
    {  
        return this.accountNumber;  
    }  
    public int queryBalance()  
    {  
        return this.balance;  
    }  
}
```

Bank Account

- balance : int
- accountNumber : String
- + deposit(int:amount):void
- + withdraw(int:amount):void
- + queryBalance():int
- + getAccountNumber():String

```
    public void deposit (int amount)  
    {  
        this.balance += amount;  
    }  
    public void withdraw (int amount)  
    {  
        this.balance -= amount;  
    }  
}
```



Class Construction

- The Class declaration

public	Class is public accessible
abstract	Class cannot be instantiated
final	Class cannot be subclassed
<i>class NameOfClass</i>	Name of the class
extends Super	Superclass of this class
implements Interfaces	Interfaces implemented by this class
{ ClassBody }	



Class Construction

- The Class body

- The class body contains all of the code :

- constructors,
 - declarations for the variables,
 - methods.

« Note » Constructors are not methods. Nor are they members.





Class Construction

- The Class body
 - *Constructors*
 - All Java Classes have constructors
 - Initialize a new object of that type
 - The same name as the class
 - In the example we define a single constructor but we can define some constructors or zero constructor





Class Construction

```
public BankAccount(String accountNumber)
{
    this.accountNumber=accountNumber;
}
public BankAccount(String accountNumber,int initialBalance)
{
    this.accountNumber=accountNumber;
    this.balance=initialBalance;
}
```





Class Construction

- The Class body
 - *Constructors*
 - *Access specifiers for constructors*
 - *private*
 - *protected*
 - *public*
 - *package (default)*





Class Construction

- The Class body
 - *Declaring Member Variables*
 - A class's state is represented by its member variables

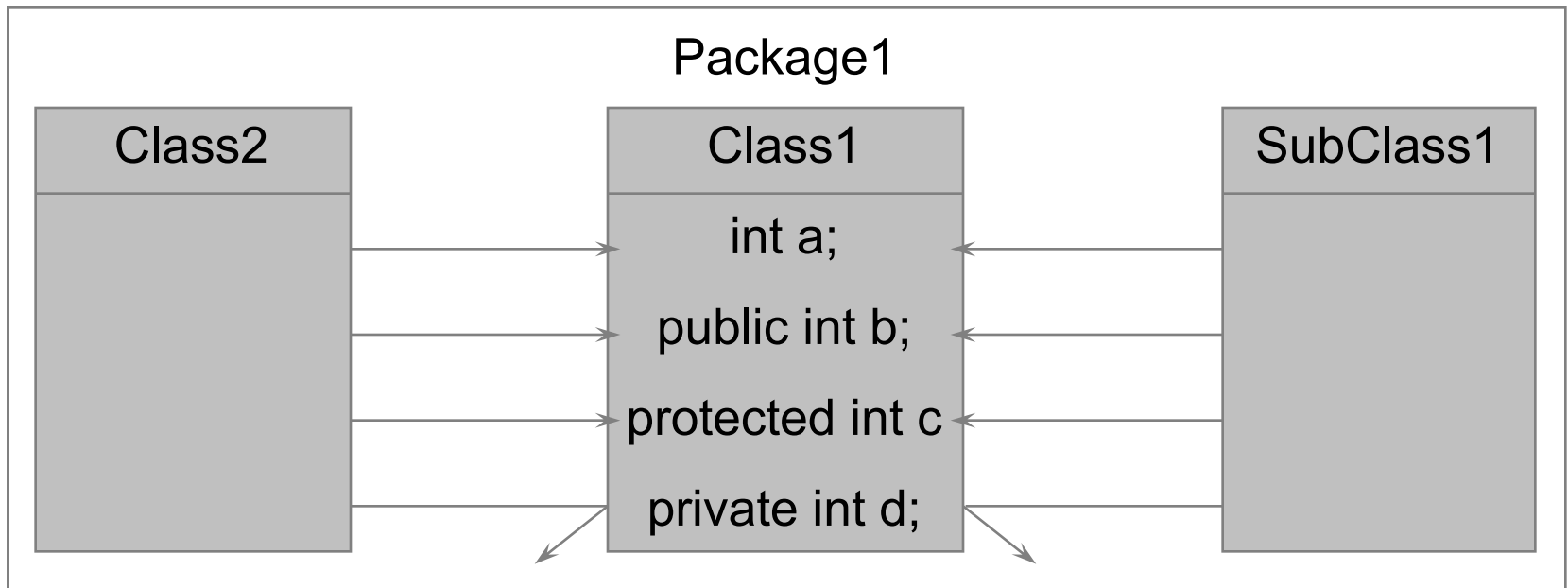
```
public class BankAccount
{
    private int balance;
    private String accountNumber;
    ...
}
```

<i>Declaration of Member Variable</i>	
accessLevel	Indicates the access level for this member
static	Declares a class member
final	Indicates that is a constant
transient	This variable is transient
volatile	This variable is volatile
<i>type name</i>	the type and the name of the variable



Class Construction

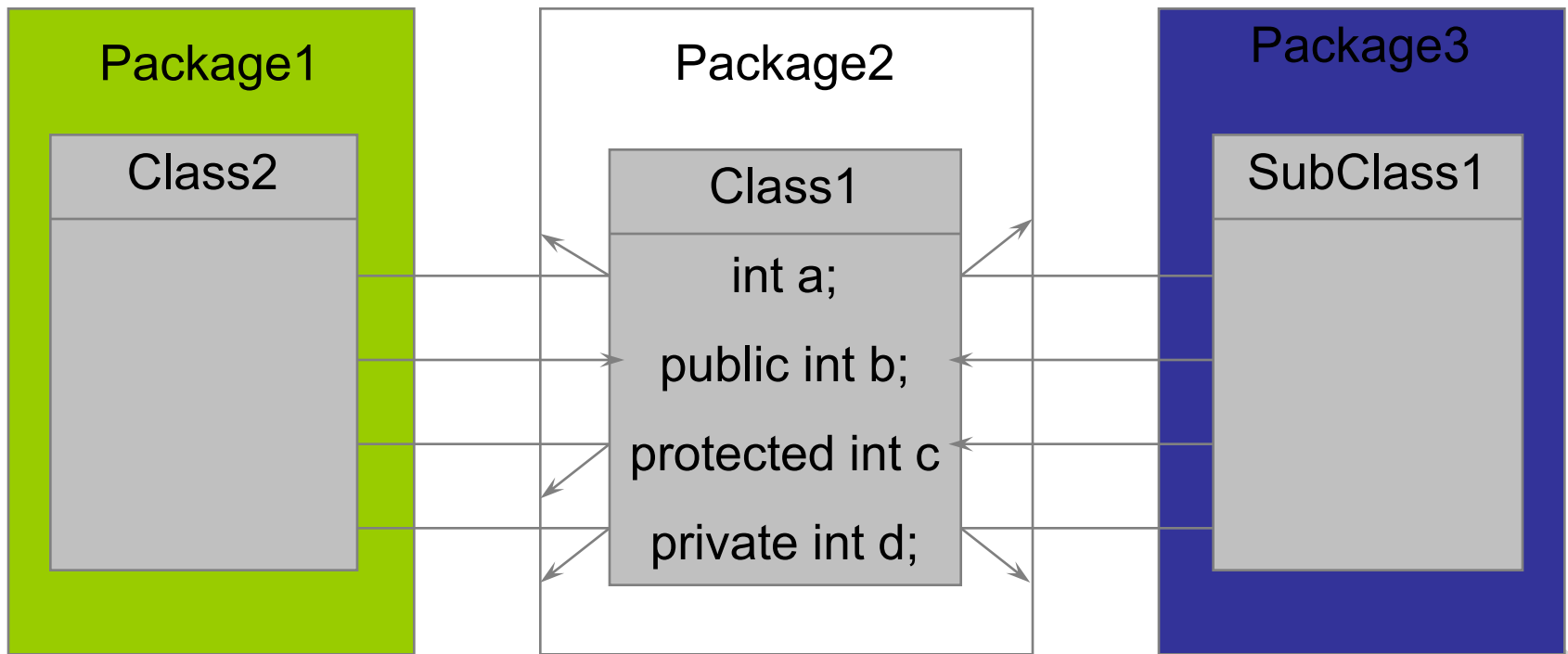
- The Class body
 - *Declaring Member Variables*
 - *AccessLevel*





Class Construction

- The Class body
 - *Declaring Member Variables*
 - *AccessLevel*





Class Construction

- The Class body
 - *Declaring Member Variables*
 - *static*
 - *final*
 - *transient*
 - *volatile*
 - *type*
 - *name*





Class Construction

- The Class body
 - *Implementing Methods*
 - As you know, objects have behavior that is implemented by its methods.

```
public class BankAccount
{
    ...
    public String getAccountNumber()
    {
        return this.accountNumber;
    }
    public void deposit (int amount)
    {
        this.balance += amount;
    }
    ...
}
```





Class Construction

- The Class body
 - *Implementing Methods*
 - As you know, objects have behavior that is implemented by its methods.

Method Declaration

```
public class BankAccount
{
    ...
    public String getAccountNumber()
    {
        return this.accountNumber;
    }
    public void deposit (int amount)
    {
        this.balance += amount;
    }
    ...
}
```



Class Construction

- The Class body
 - *Implementing Methods*
 - As you know, objects have behavior that is implemented by its methods.

Method Body

```
public class BankAccount
{
    ...
    public String getAccountNumber()
    {
        return this.accountNumber;
    }
    public void deposit (int amount)
    {
        this.balance += amount;
    }
    ...
}
```





Class Construction

- The Class body
 - *Implementing Methods*
 - Method Declaration

Elements of a Method Declaration

accessLevel	Indicates the access level for this method
static	Declares a class method
abstract	This method is not implemented
final	Method cannot be overridden
native	Method implemented in another language
Synchronized	Method requires a monitor to run
<i>returntype methodName</i>	the return type and the method name
(paramlist)	the list of arguments
throws exceptions	The exceptions thrown by this method



Class Construction

- The Class body
 - *Implementing Methods*
 - Returning a Value from a method
 - If method's return type is not **void** → use the **return** operator to return the value

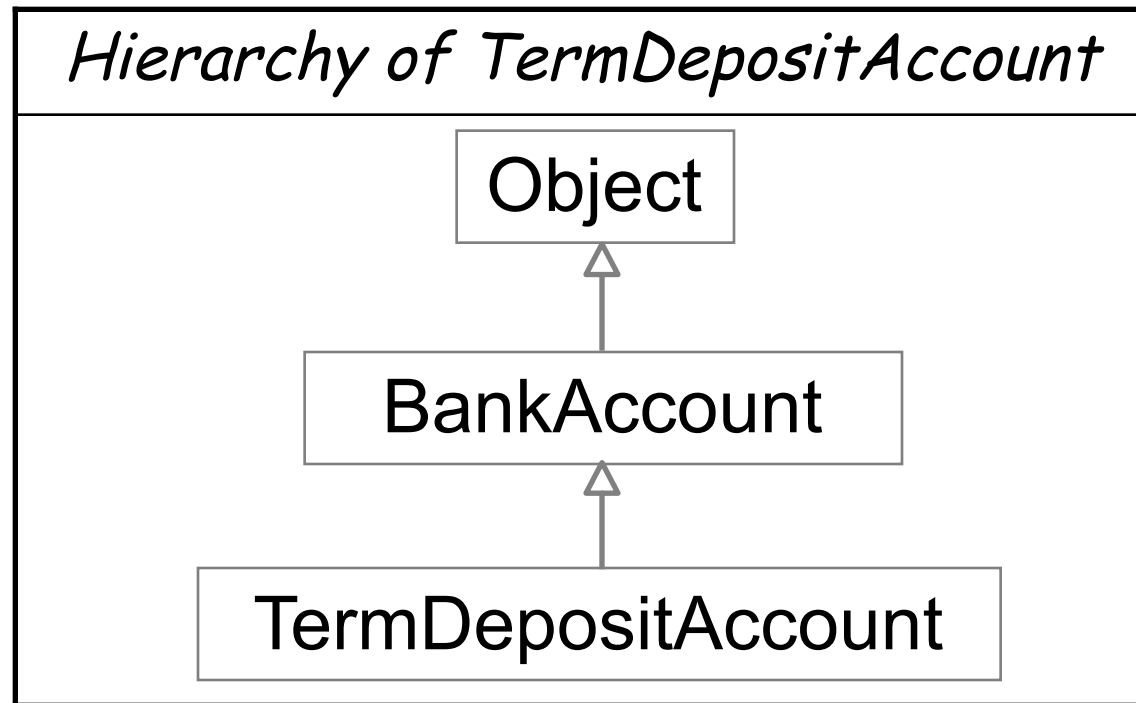
```
public class BankAccount
{
    ...
    public String getAccountNumber()
    {
        return this.accountNumber;
    }
    public void deposit (int amount)
    {
        this.balance += amount;
    }
    ...
}
```





Class Construction

- The Class body
 - *Implementing Methods*
 - Returning a Value from a method



- the getAccountNumber method can return a TermDepositAccount but not a Object





Class Construction

- The Class body

- *Implementing Methods*

- Passing information into a Method

- declare the type and name for each arguments in the **method signature**

```
public int computePerimeter(int side1, int
side2, int side3)
{
    return side1 + side2 + side3;
}
```

- Argument types are valid Java data types like : primitive data types, reference data types (objects, arrays, interfaces)
 - Argument names is used to refer to the item into the method body.
 - if a method argument have the same name of a class's member the argument **hide** the class's member
-



Class Construction

- The Class body
 - *Implementing Methods*
 - Method Body
 - Local Variables
 - Within method body you can declare more variables

```
public int computePerimeter(int side1, int side2, int side3)
{
    int perimeter = side1 + side 2 + side3;
    return perimeter;
}
```





Class Construction

- The Class body
 - *The keywords **this** & **super***
 - Use **this** to refer members in the current object
 - Use **super** to refer members in the superclass that the current class has hidden or overridden





Contents

- Class Construction
 - Class Declaration
 - Class Body
- Understanding Instance and Class Members
 - Instance variables & instance methods
 - Class variable & class methods
 - Objects vs. Classes



Understanding Instance and Class Members

- Instance variables & instance methods
 - The values for instance variables are provided by each instance of the class.
 - When you create a class you must instantiate it before you can use it.
 - You can now invoke instance methods of this object
 - Instances of the same class share the same **instance method implementations**, which **reside in the class itself**



Understanding Instance and Class Members

- Class variable & class methods
 - Classes can also define class variables and class methods
 - You don't have to instantiate a class to use its class methods and variables
 - class methods use only class variables (not instance methods or variables)
 - Single copy of all class variables when the program uses this class the first time



Understanding Instance and Class Members

- Class variable & class methods

BankAccount
-balance:int -accountNumber:String <u>-interestRate:int</u>
+ deposit(int amount):void + withdraw(int amout):void +queryBalance():int + getAccountNumber():String + <u>setInterestRate(int rate):void</u> + <u>getInterestRate():int</u>

Understanding Instance and Class Members

- Objects vs. Classes

- In the real world, the classes are not themselves the objects that they describe

- A blueprint of a bank account is not a bank account.

- Classes contain :

- description of the \neq variables and methods;
 - values of the class variables;
 - link to their superclass.

- Objects contain :

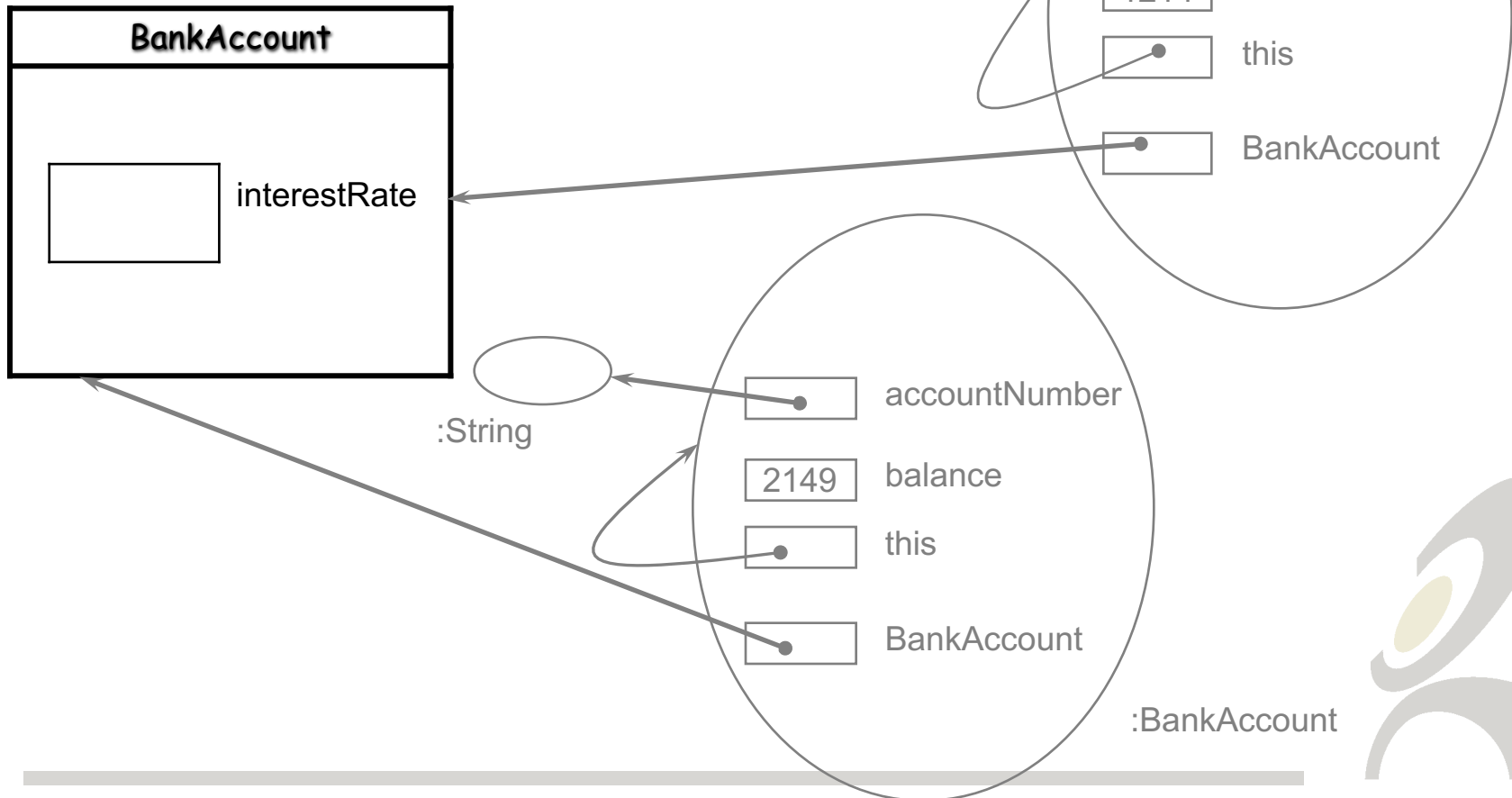
- the actual value of the instance variables;
 - a link to their class type



Understanding Instance and Class Members

:BankAccount

- Objects vs. Classes





Arrays

- Definition
 - Structure holding multiple values (same type)
 - Length established when created and fixed
 - *Array element* is one of value
 - *Array element* is accessed by its position





Arrays

An array is like a tray of cups

- 1 Declare an int array variable. An array variable is a remote control to an array object.

```
int[] nums;
```

- 2 Create a new int array with a length of 7, and assign it to the previously-declared `int[]` variable `nums`

```
nums = new int[7];
```

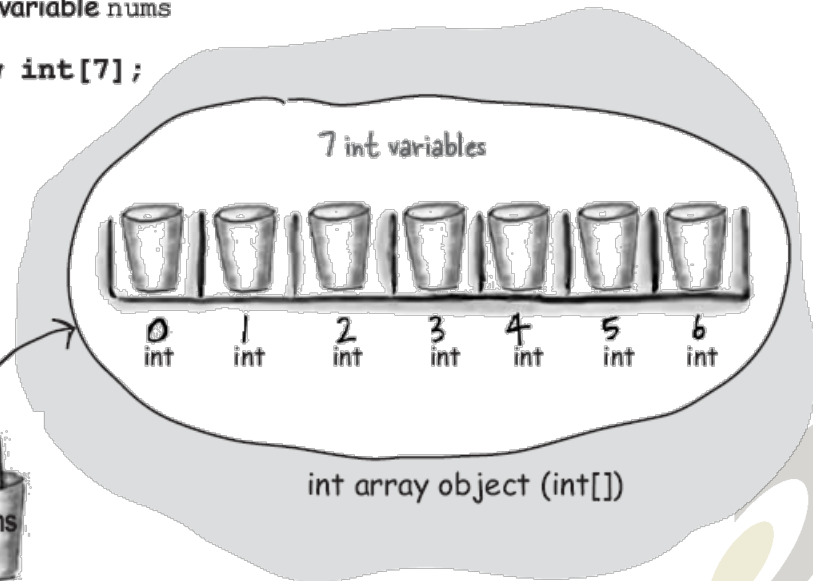
- 3 Give each element in the array an int value. Remember, elements in an int array are just int variables.

7 int variables

```
nums[0] = 6;  
nums[1] = 19;  
nums[2] = 44;  
nums[3] = 42;  
nums[4] = 10;  
nums[5] = 20;  
nums[6] = 1;
```



int[]



Notice that the array itself is an object, even though the 7 elements are primitives.



Arrays

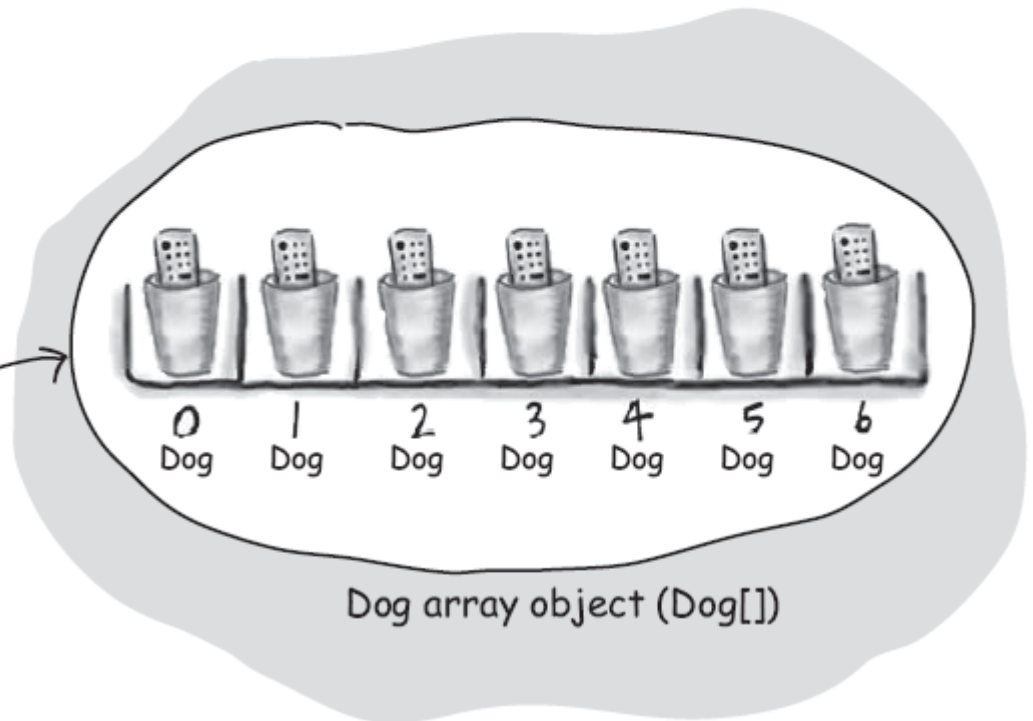
Make an array of Dogs

- 1 Declare a Dog array variable
`Dog[] pets;`
- 2 Create a new Dog array with a length of 7, and assign it to the previously-declared `Dog[]` variable `pets`

```
pets = new Dog[7];
```

What's missing?

Dogs! We have an array of Dog *references*, but no actual Dog *objects*!





Arrays

- 3** Create new Dog objects, and assign them to the array elements.
Remember, elements in a Dog array are just Dog reference variables. We still need Dogs!

```
pets[0] = new Dog();  
pets[1] = new Dog();
```



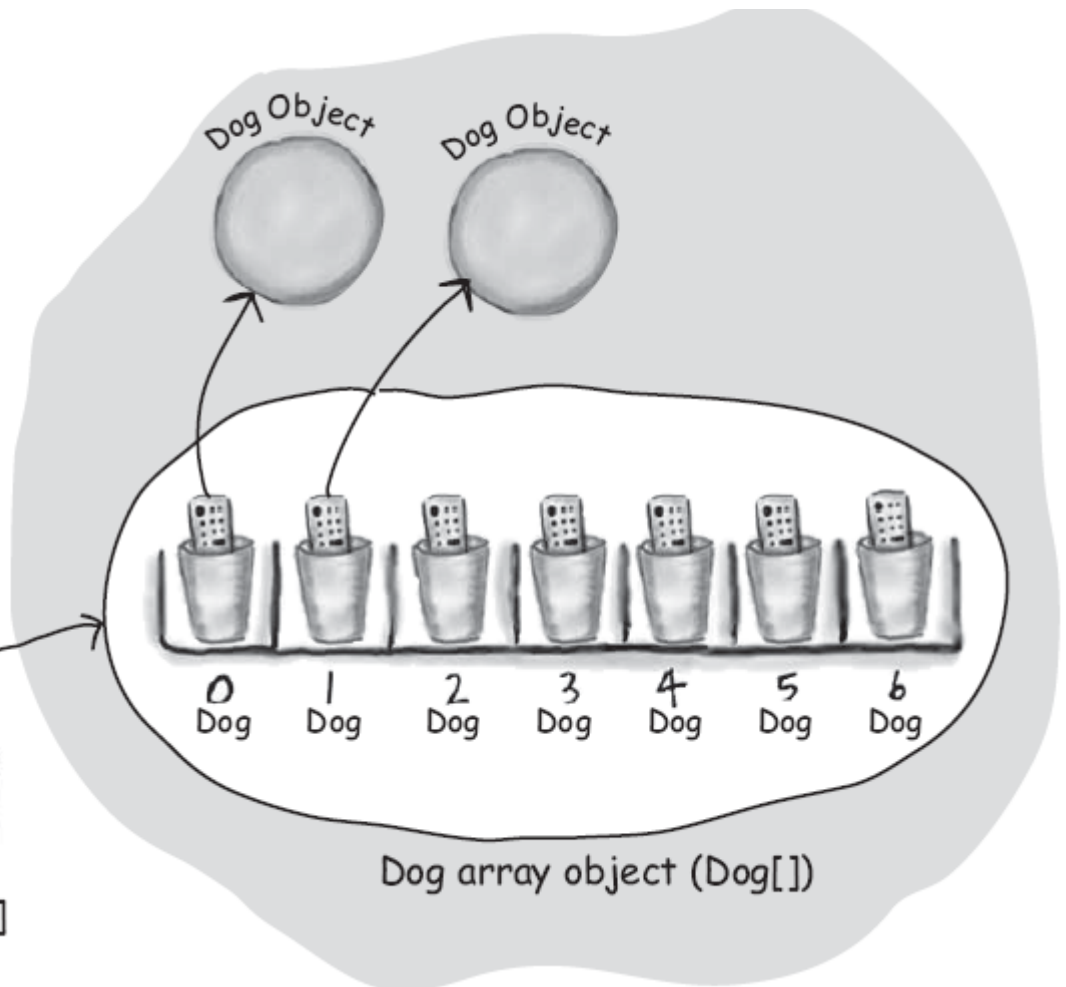
Sharpen your pencil

What is the current value of `pets[2]`? _____

What code would make `pets[3]` refer to one of the two existing Dog objects?



`Dog[]`





Arrays

- Creating and Using Arrays

```
public class ArrayDemo {  
    public static void main(String[] args) {  
        int[] anArray;  
        anArray = new int[10];  
  
        for (int i = 0; i < anArray.length; i++) {  
            anArray[i] = i;  
            System.out.print(anArray[i] + " ");  
        }  
        System.out.println();  
    }  
}
```



Arrays

- Getting the Size of an Array
 - `anArray.length`
- Array Initializers
 - `boolean[] answers = {true, false, true, true};`





Questions ??

