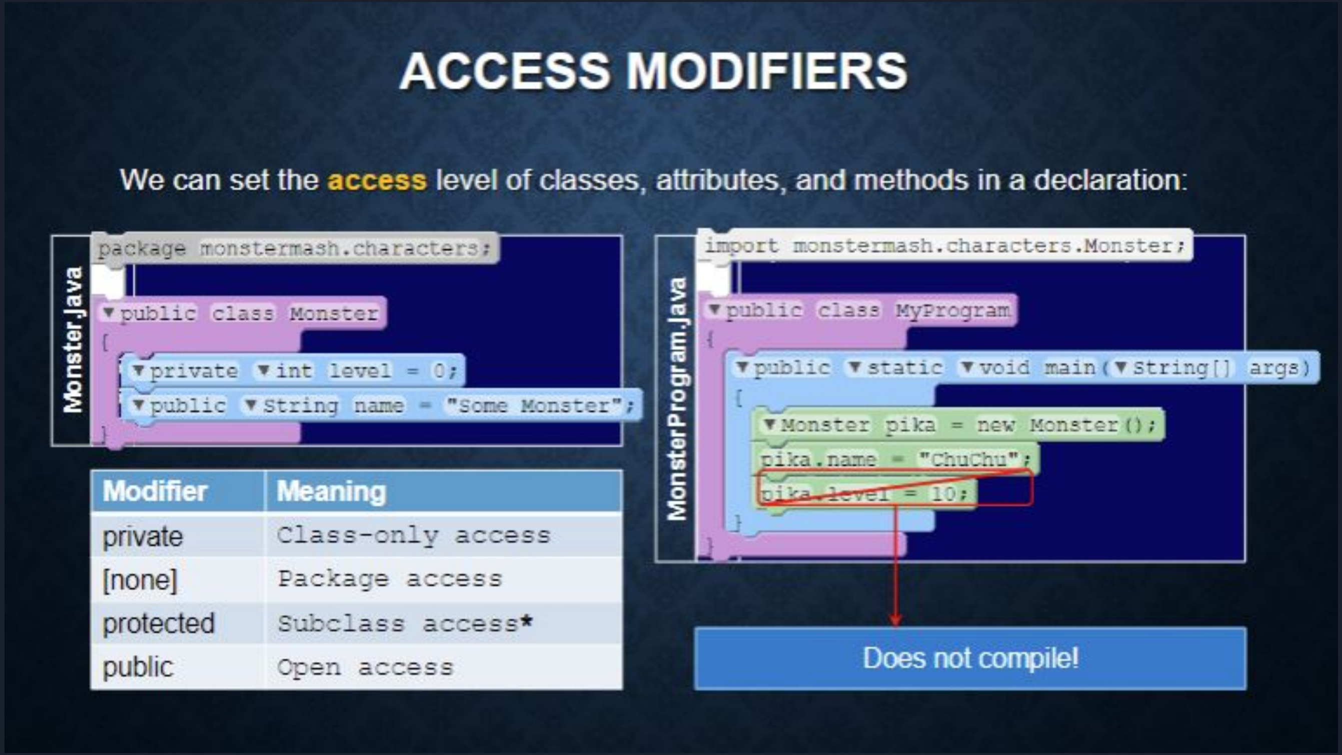
Exam 2 Review

- MCQ (including fill in blank) (9 * 5 points)
- Find Errors in a Program (5 + 6 points) - Exception and Arrays
- Predict the output of a program (24 points) - Array, Class, Inheritance/Polymorphism
- Coding Question (2 * 10 points) - Array and Class definition
- 1 or 2 Extra Credits (10 points)

What is a class?

Design for an object with attributes and methods



ACCESS MODIFIERS

We can set the **access** level of classes, attributes, and methods in a declaration:

Monster.java

```
package monstermash.characters;

public class Monster
{
    private int level = 0;
    public String name = "Some Monster";
}
```

MonsterProgram.java

```
import monstermash.characters.Monster;

public class MyProgram
{
    public static void main(String[] args)
    {
        Monster pika = new Monster();
        pika.name = "ChuChu";
        pika.level = 10;
    }
}
```

Does not compile!

Modifier	Meaning
private	Class-only access
[none]	Package access
protected	Subclass access*
public	Open access

Accessors, mutators, and constructors

Accessors/Getters- allow us to GET information, NOT edit

Mutators/Setters- allow us to CHANGE information

Constructors - create an instance of a class

Overloading

Allows us to initialize an object multiple ways

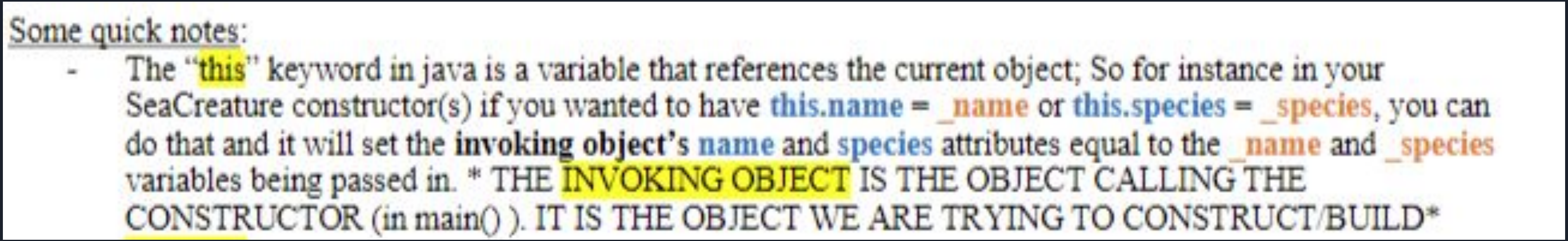
- Want a default constructor with no user input? Sure!
- Add the ability to edit one field? Add another constructor!

Overloading methods works the same way! Different parameters are the only requirement.

- Can’t have two methods with the same name that each take in one integer value

THIS

This references a specific instance of a class. Helps specialize the variable so the program knows we are talking about the current instance.



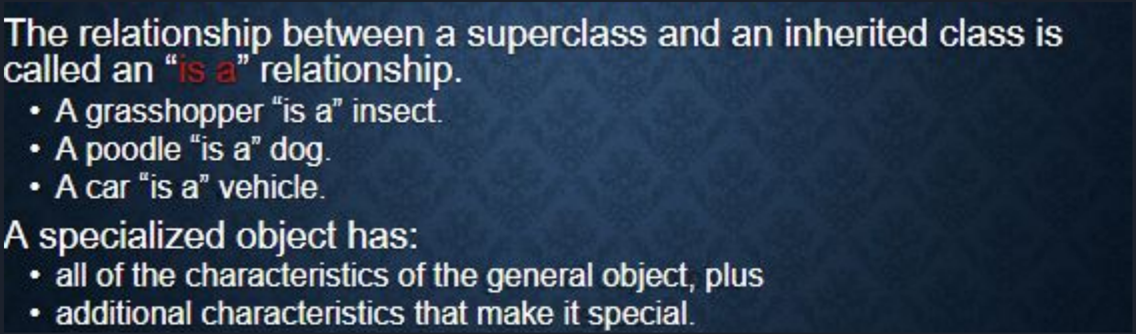
Some quick notes:

- The "this" keyword in java is a variable that references the current object; So for instance in your SeaCreature constructor(s) if you wanted to have this.name = _name or this.species = _species, you can do that and it will set the **invoking object's** name and species attributes equal to the _name and _species variables being passed in. * THE INVOKING OBJECT IS THE OBJECT CALLING THE CONSTRUCTOR (in main()). IT IS THE OBJECT WE ARE TRYING TO CONSTRUCT/BUILD*

Inheritance, what is it?

This allows us to take more specific classes and absorb more generic information from a superclass

- subclass (child) - the class that inherits from another class
- superclass (parent) - the class being inherited from

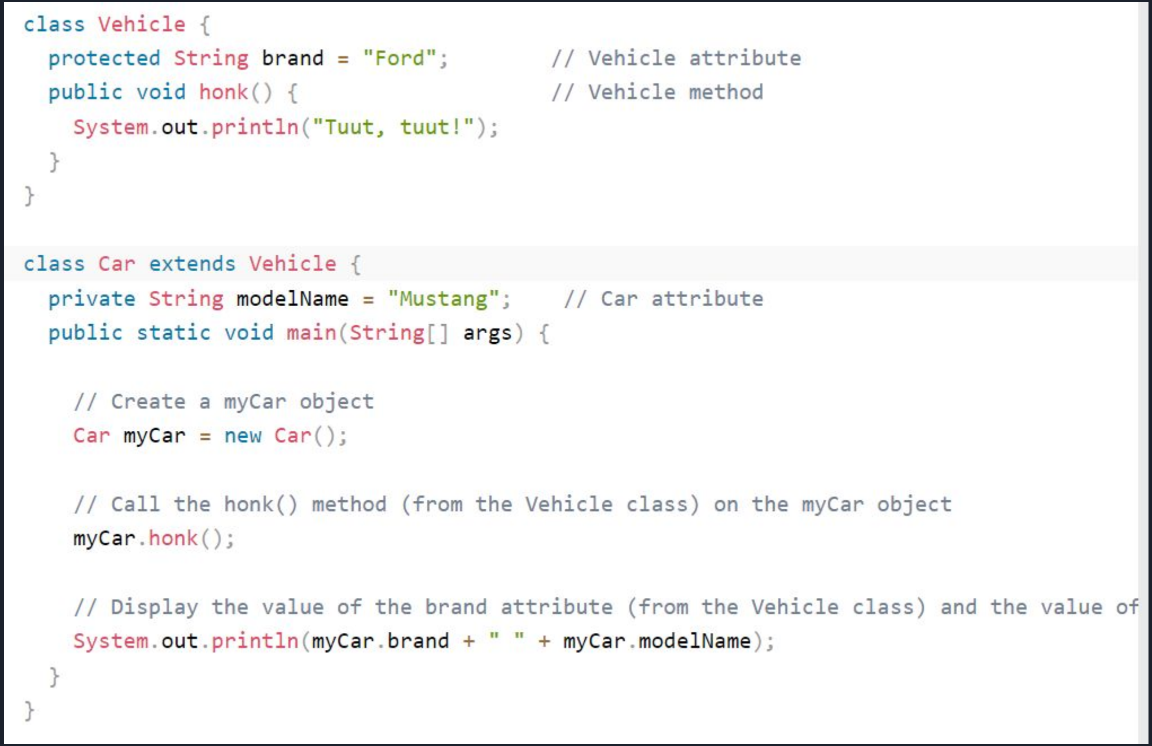

The relationship between a superclass and an inherited class is called an “is a” relationship.

- A grasshopper “is a” insect.
- A poodle “is a” dog.
- A car “is a” vehicle.

A specialized object has:

- all of the characteristics of the general object, plus
- additional characteristics that make it special.

Inheritance



```
class Vehicle {
  protected String brand = "Ford";        // Vehicle attribute
  public void honk() {                    // Vehicle method
    System.out.println("Tuut, tuut!");
  }
}

class Car extends Vehicle {
  private String modelName = "Mustang";    // Car attribute
  public static void main(String[] args) {

    // Create a myCar object
    Car myCar = new Car();

    // Call the honk() method (from the Vehicle class) on the myCar object
    myCar.honk();

    // Display the value of the brand attribute (from the Vehicle class) and the value of
    System.out.println(myCar.brand + " " + myCar.modelName);
  }
}
```

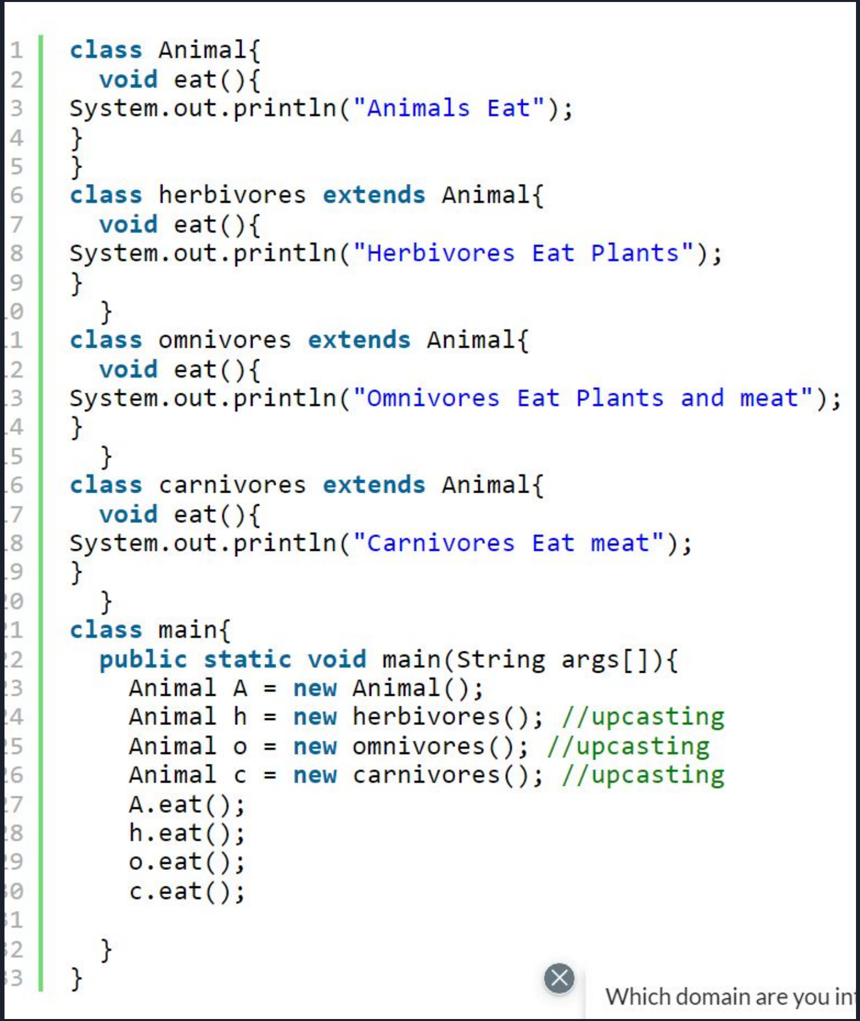
Runtime Polymorphism

What is included?

- Inheritance
- Method overriding
- Superclass reference

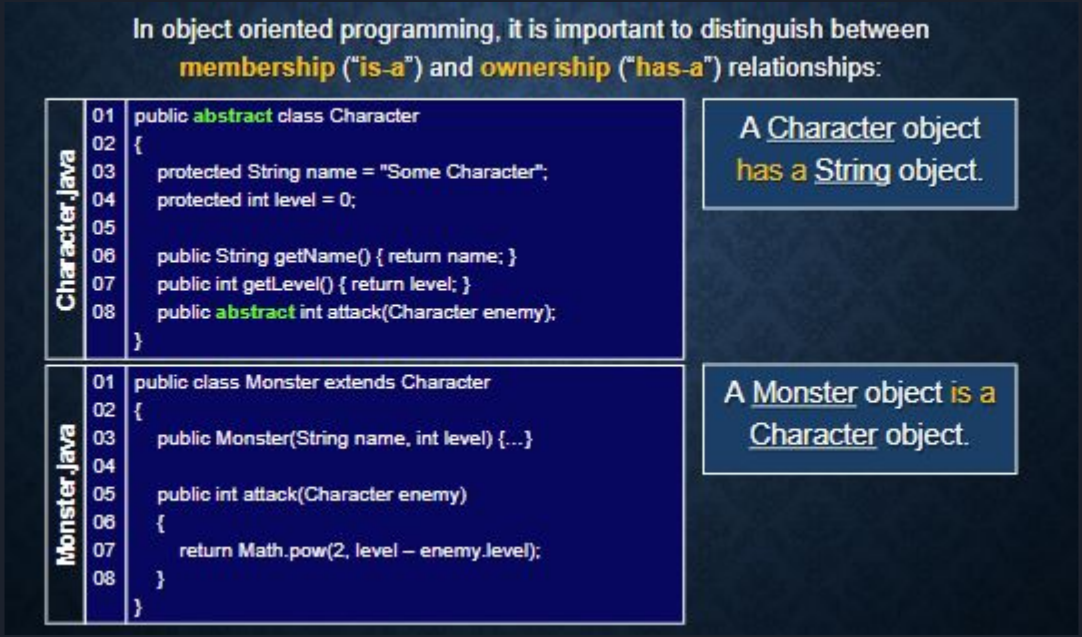
Java uses dynamic method lookup

- Dynamic method lookup is the process of determining which method definition a method signature denotes during runtime, based on the type of the object.
- Method to be called is determined by the object, not the type of reference variable



```java
class Animal{
  void eat(){
System.out.println("Animals Eat");
}
}
class herbivores extends Animal{
  void eat(){
System.out.println("Herbivores Eat Plants");
}
  }
class omnivores extends Animal{
  void eat(){
System.out.println("Omnivores Eat Plants and meat");
}
  }
class carnivores extends Animal{
  void eat(){
System.out.println("Carnivores Eat meat");
}
  }
class main{
  public static void main(String args[]){
    Animal A = new Animal();
    Animal h = new herbivores(); //upcasting
    Animal o = new omnivores(); //upcasting
    Animal c = new carnivores(); //upcasting
    A.eat();
    h.eat();
    o.eat();
    c.eat();

  }
}
```

Is-A vs Has-A


In object oriented programming, it is important to distinguish between membership ("is-a") and ownership ("has-a") relationships:

Character.Java

```
public abstract class Character
{
   protected String name = "Some Character";
   protected int level = 0;

   public String getName() { return name; }
   public int getLevel() { return level; }
   public abstract int attack(Character enemy);
}
```

A Character object has a String object.

Monster.Java

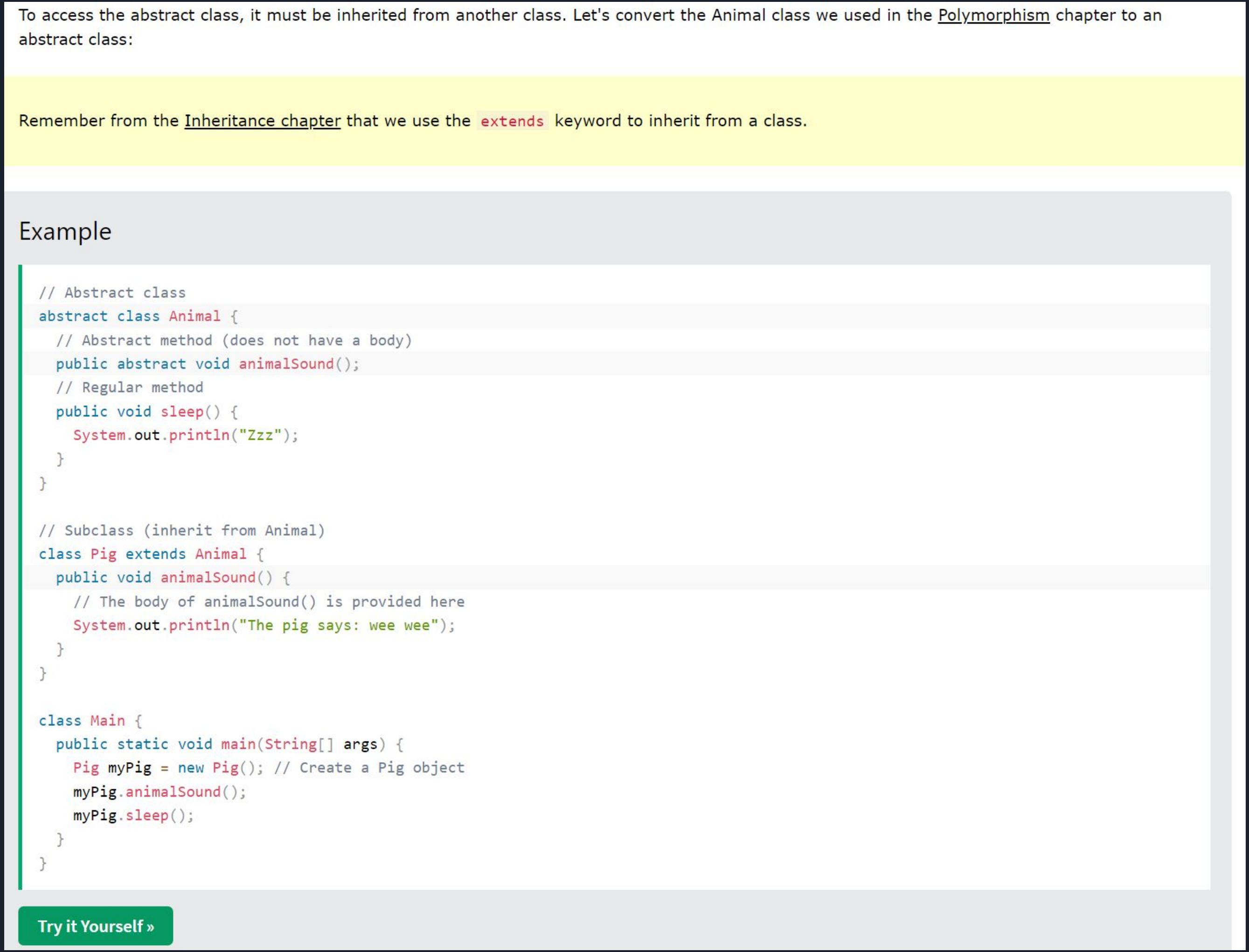
```
public class Monster extends Character
{
   public Monster(String name, int level) {...}

   public int attack(Character enemy)
   {
      return Math.pow(2, level – enemy.level);
   }
}
```

A Monster object is a Character object.

Abstraction

The `abstract` keyword is a non-access modifier, used for classes and methods:

- Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
- Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

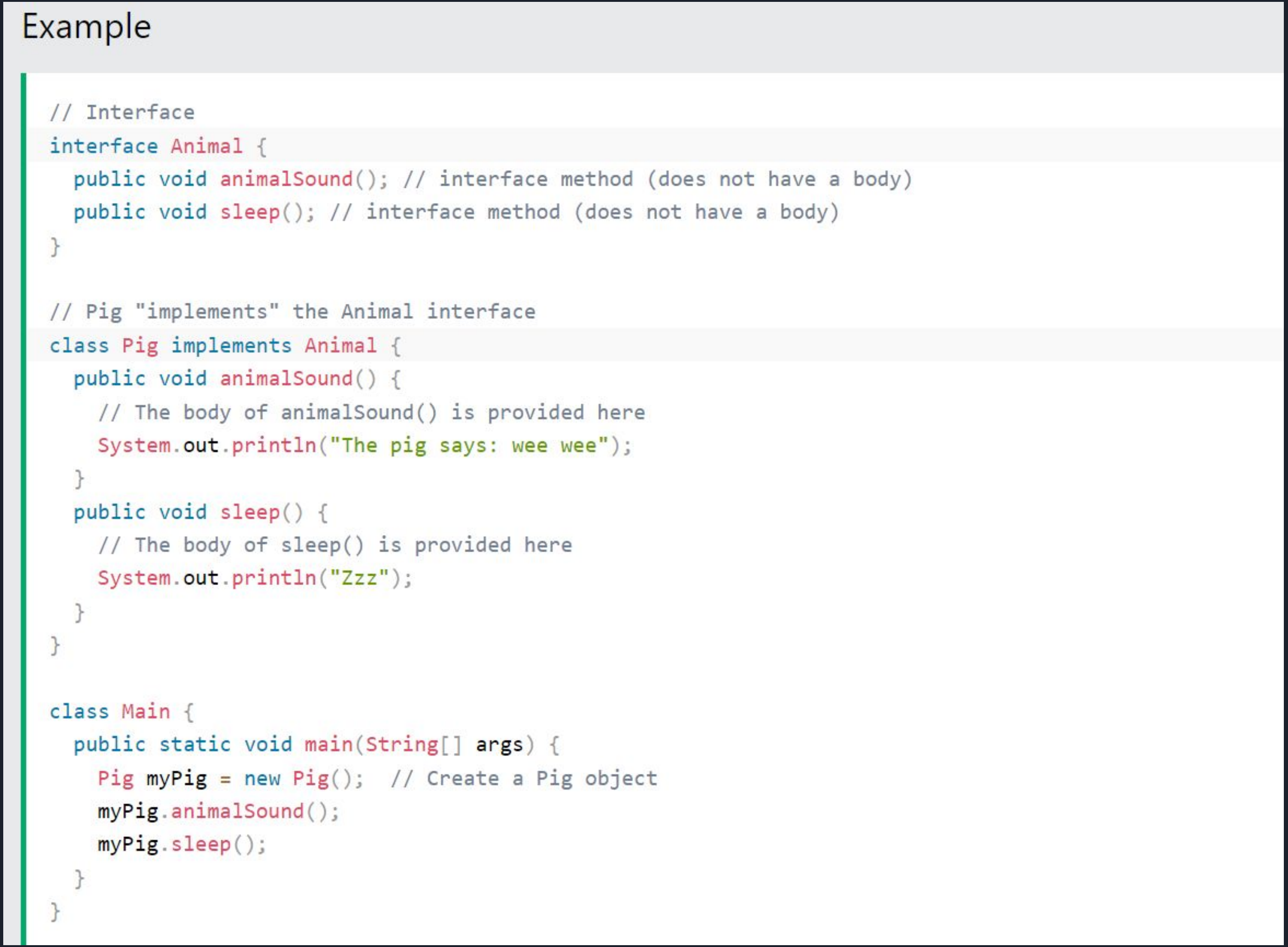


To access the abstract class, it must be inherited from another class. Let's convert the Animal class we used in the Polymorphism chapter to an abstract class:

Remember from the Inheritance chapter that we use the `extends` keyword to inherit from a class.

Example

```java
// Abstract class
abstract class Animal {
  // Abstract method (does not have a body)
  public abstract void animalSound();
  // Regular method
  public void sleep() {
    System.out.println("Zzz");
  }
}

// Subclass (inherit from Animal)
class Pig extends Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig(); // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

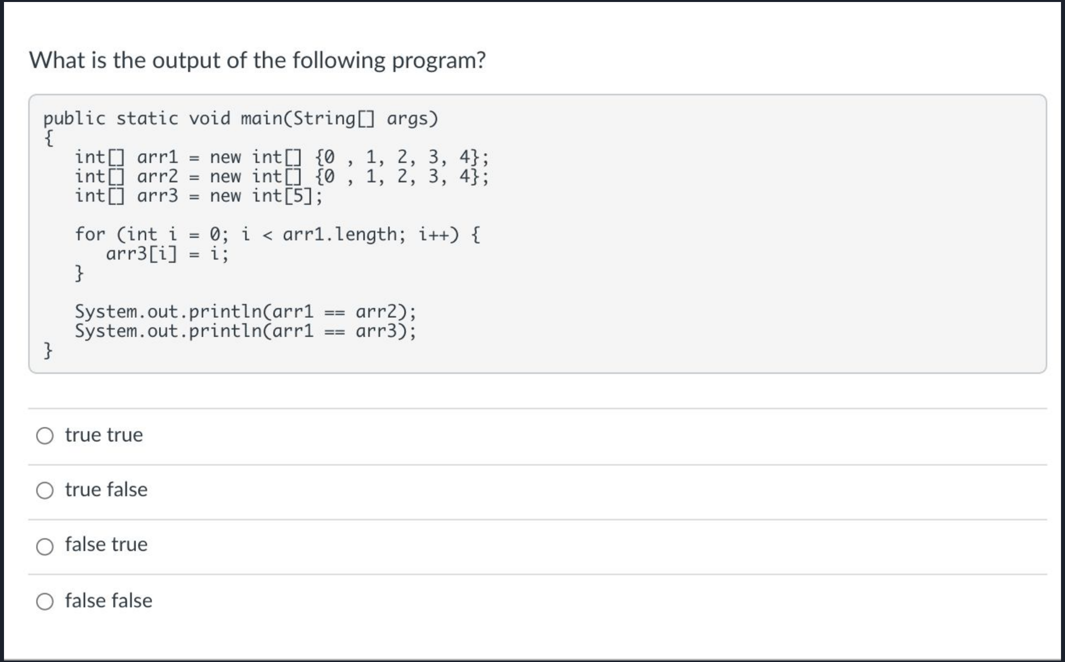


Example

```
// Interface
interface Animal {
  public void animalSound(); // interface method (does not have a body)
  public void sleep(); // interface method (does not have a body)
}

// Pig "implements" the Animal interface
class Pig implements Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
  public void sleep() {
    // The body of sleep() is provided here
    System.out.println("Zzz");
  }
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig();  // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

PRACTICE QUESTIONS:

4B Q1:

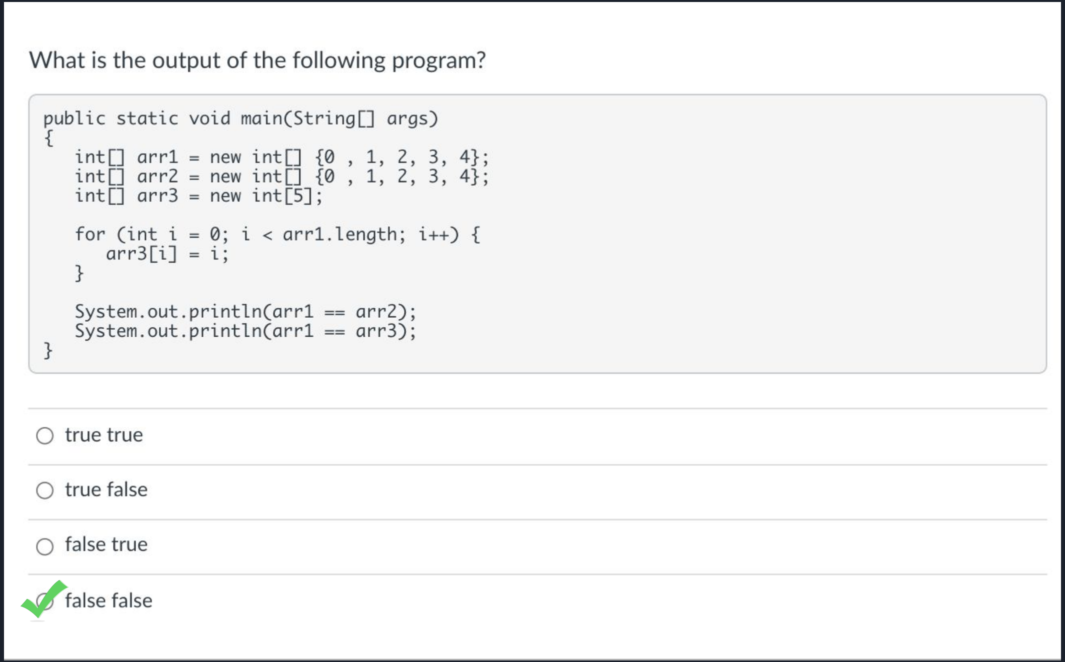


What is the output of the following program?

```
public static void main(String[] args)
{
   int[] arr1 = new int[] {0 , 1, 2, 3, 4};
   int[] arr2 = new int[] {0 , 1, 2, 3, 4};
   int[] arr3 = new int[5];

   for (int i = 0; i < arr1.length; i++) {
      arr3[i] = i;
   }

   System.out.println(arr1 == arr2);
   System.out.println(arr1 == arr3);
}
```

- ○ true true
- ○ true false
- ○ false true
- ○ false false

4B A1:



What is the output of the following program?

```
public static void main(String[] args)
{
   int[] arr1 = new int[] {0 , 1, 2, 3, 4};
   int[] arr2 = new int[] {0 , 1, 2, 3, 4};
   int[] arr3 = new int[5];

   for (int i = 0; i < arr1.length; i++) {
      arr3[i] = i;
   }

   System.out.println(arr1 == arr2);
   System.out.println(arr1 == arr3);
}
```

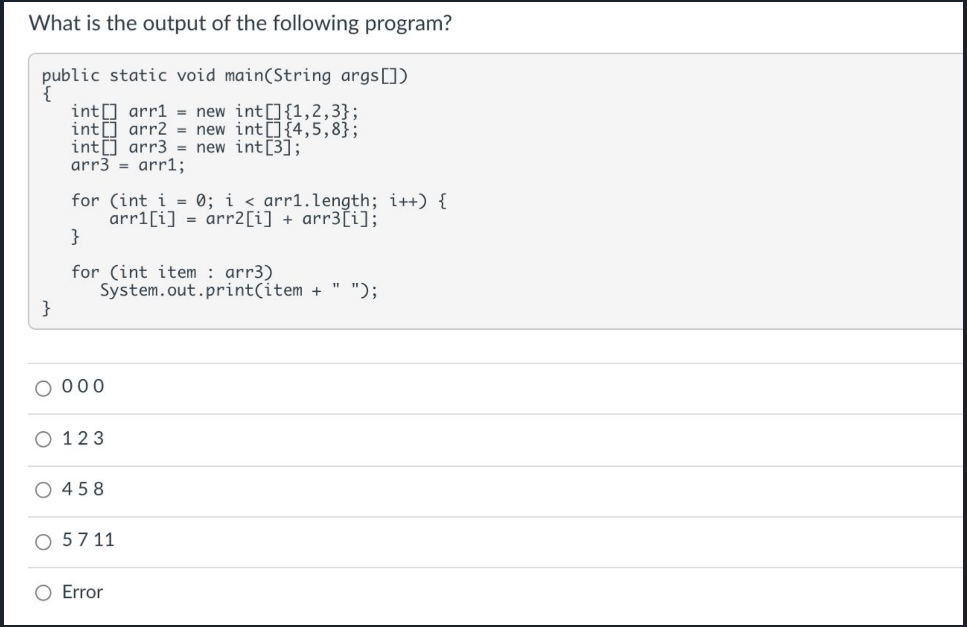
○ true true

○ true false

○ false true

[✓ false false]

4B Q2:

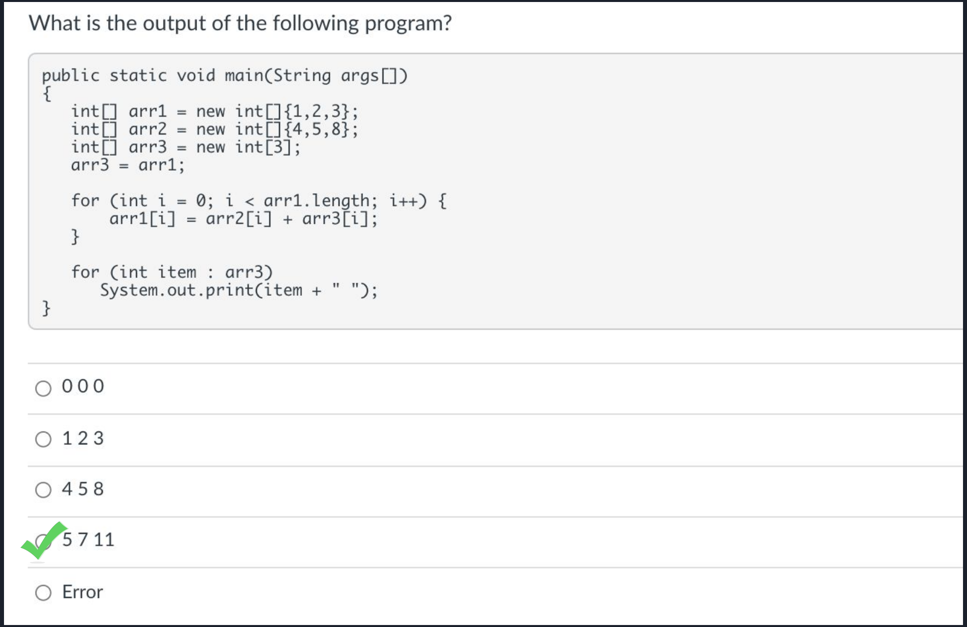


What is the output of the following program?

```
public static void main(String args[])
{
   int[] arr1 = new int[]{1,2,3};
   int[] arr2 = new int[]{4,5,8};
   int[] arr3 = new int[3];
   arr3 = arr1;

   for (int i = 0; i < arr1.length; i++) {
       arr1[i] = arr2[i] + arr3[i];
   }

   for (int item : arr3)
      System.out.print(item + " ");
}
```

- ○ 0 0 0
- ○ 1 2 3
- ○ 4 5 8
- ○ 5 7 11
- ○ Error

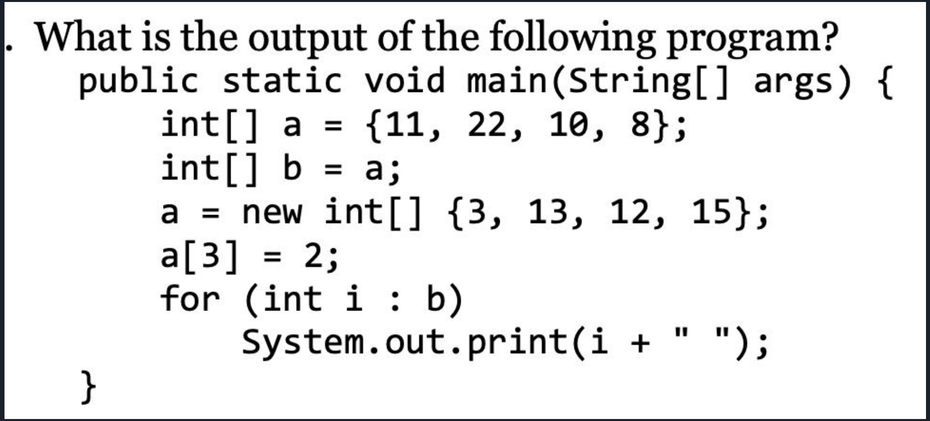
4B A2:


What is the output of the following program?

```
public static void main(String args[])
{
   int[] arr1 = new int[]{1,2,3};
   int[] arr2 = new int[]{4,5,8};
   int[] arr3 = new int[3];
   arr3 = arr1;

   for (int i = 0; i < arr1.length; i++) {
       arr1[i] = arr2[i] + arr3[i];
   }

   for (int item : arr3)
      System.out.print(item + " ");
}
```

- ○ 0 0 0
- ○ 1 2 3
- ○ 4 5 8
- [✓ 5 7 11]
- ○ Error

4B Q3:



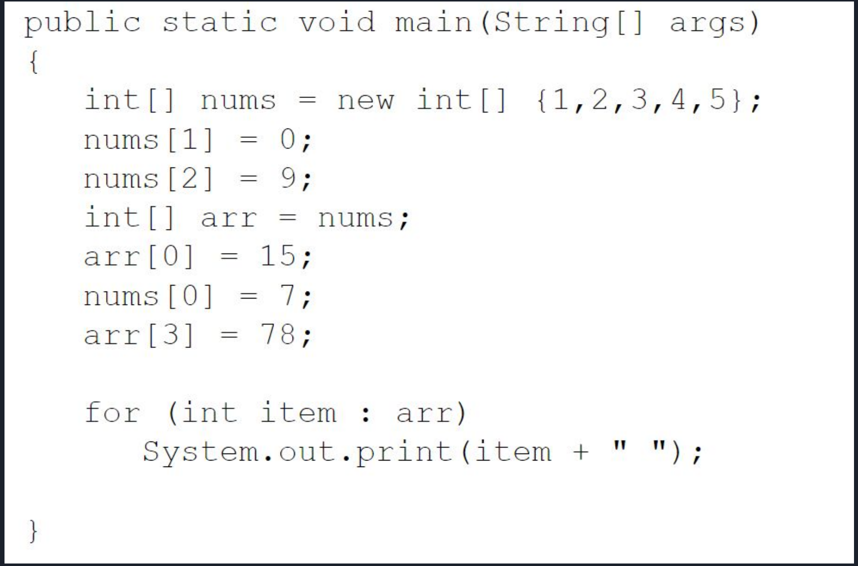
. What is the output of the following program?

```
public static void main(String[] args) {
    int[] a = {11, 22, 10, 8};
    int[] b = a;
    a = new int[] {3, 13, 12, 15};
    a[3] = 2;
    for (int i : b)
        System.out.print(i + " ");
}
```

4B A3:

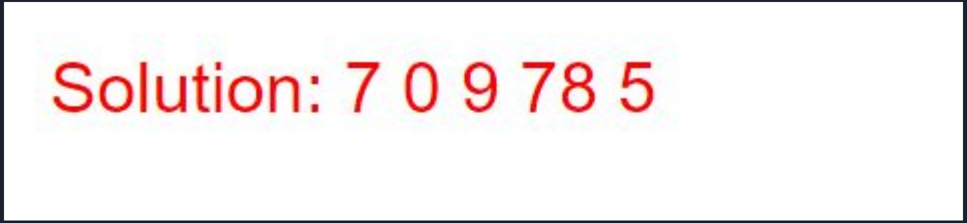
11 22 10 8

4B Q4:

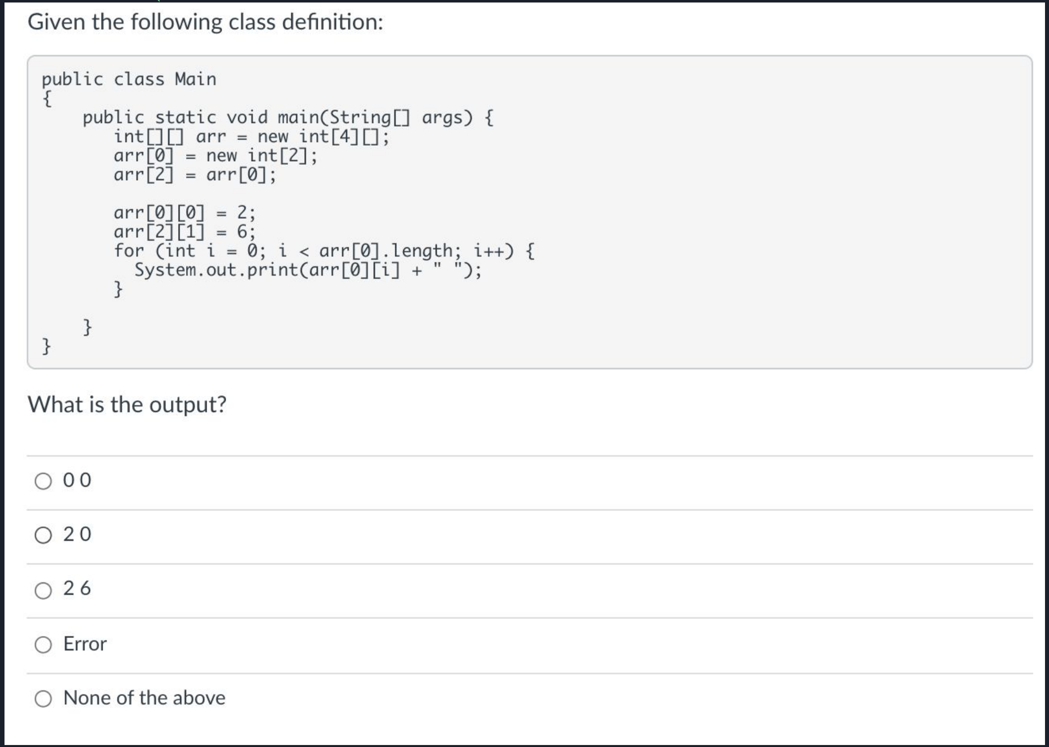


```
public static void main(String[] args)
{
   int[] nums = new int[] {1,2,3,4,5};
   nums[1] = 0;
   nums[2] = 9;
   int[] arr = nums;
   arr[0] = 15;
   nums[0] = 7;
   arr[3] = 78;

   for (int item : arr)
      System.out.print(item + " ");

}
```

4B A4:


Solution: 7 0 9 78 5

4B Q5:



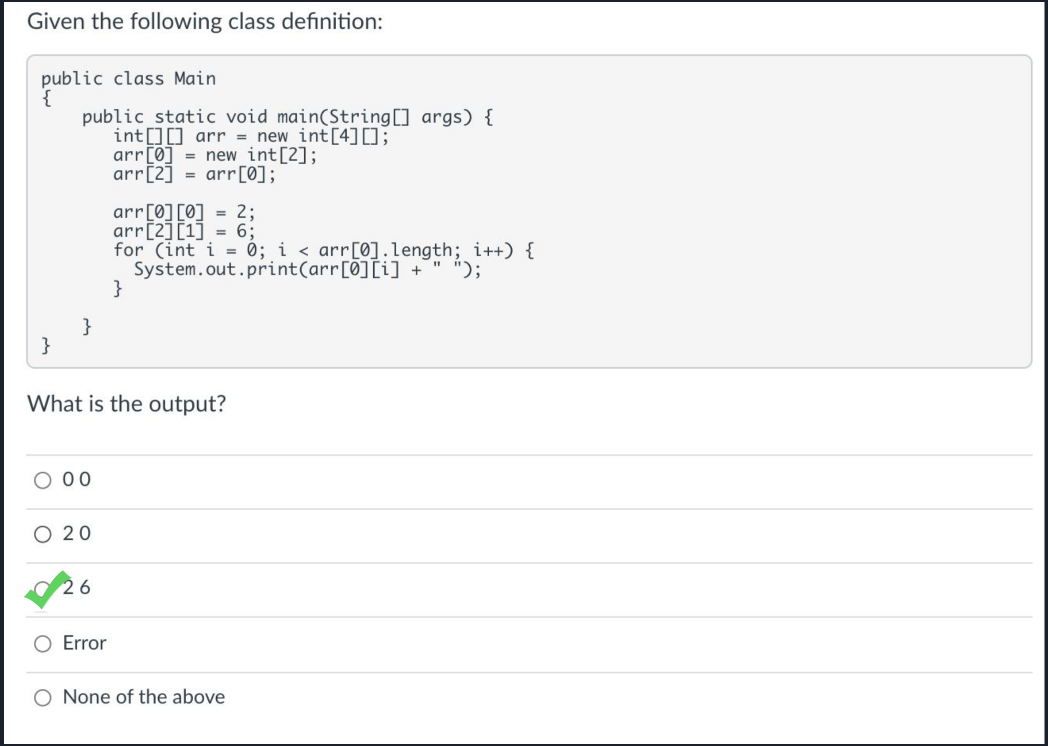
Given the following class definition:

```
public class Main
{
    public static void main(String[] args) {
       int[][] arr = new int[4][];
       arr[0] = new int[2];
       arr[2] = arr[0];

       arr[0][0] = 2;
       arr[2][1] = 6;
       for (int i = 0; i < arr[0].length; i++) {
         System.out.print(arr[0][i] + " ");
       }

    }
}
```

What is the output?

- ○ 0 0
- ○ 2 0
- ○ 2 6
- ○ Error
- ○ None of the above

4B A5:



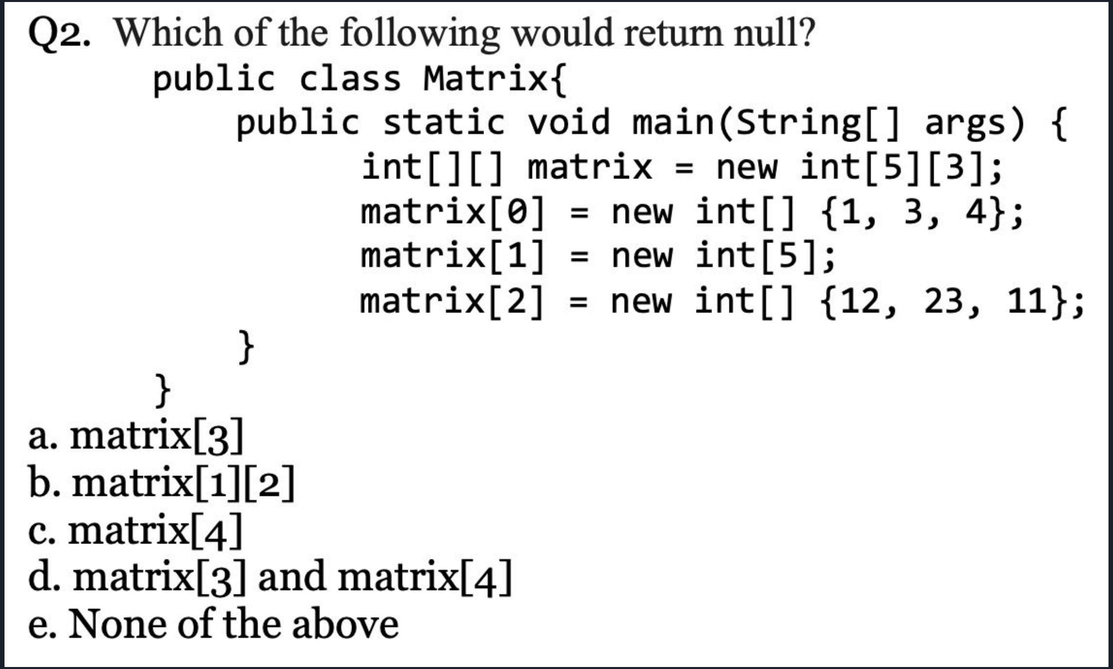
Given the following class definition:

```
public class Main
{
    public static void main(String[] args) {
       int[][] arr = new int[4][];
       arr[0] = new int[2];
       arr[2] = arr[0];

       arr[0][0] = 2;
       arr[2][1] = 6;
       for (int i = 0; i < arr[0].length; i++) {
         System.out.print(arr[0][i] + " ");
       }

    }
}
```

What is the output?

- 0 0
- 2 0
- [✓ 2 6]
- Error
- None of the above

4B Q6:

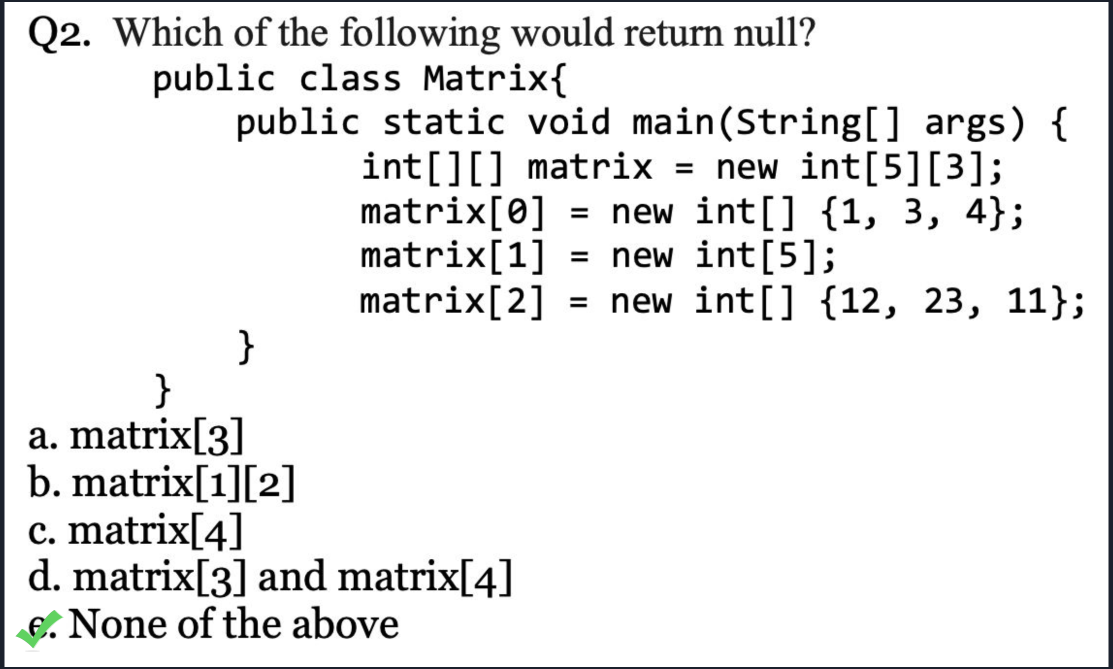


Q2. Which of the following would return null?

```
public class Matrix{
    public static void main(String[] args) {
        int[][] matrix = new int[5][3];
        matrix[0] = new int[] {1, 3, 4};
        matrix[1] = new int[5];
        matrix[2] = new int[] {12, 23, 11};
    }
}
```

a. matrix[3]
b. matrix[1][2]
c. matrix[4]
d. matrix[3] and matrix[4]
e. None of the above

4B A6:

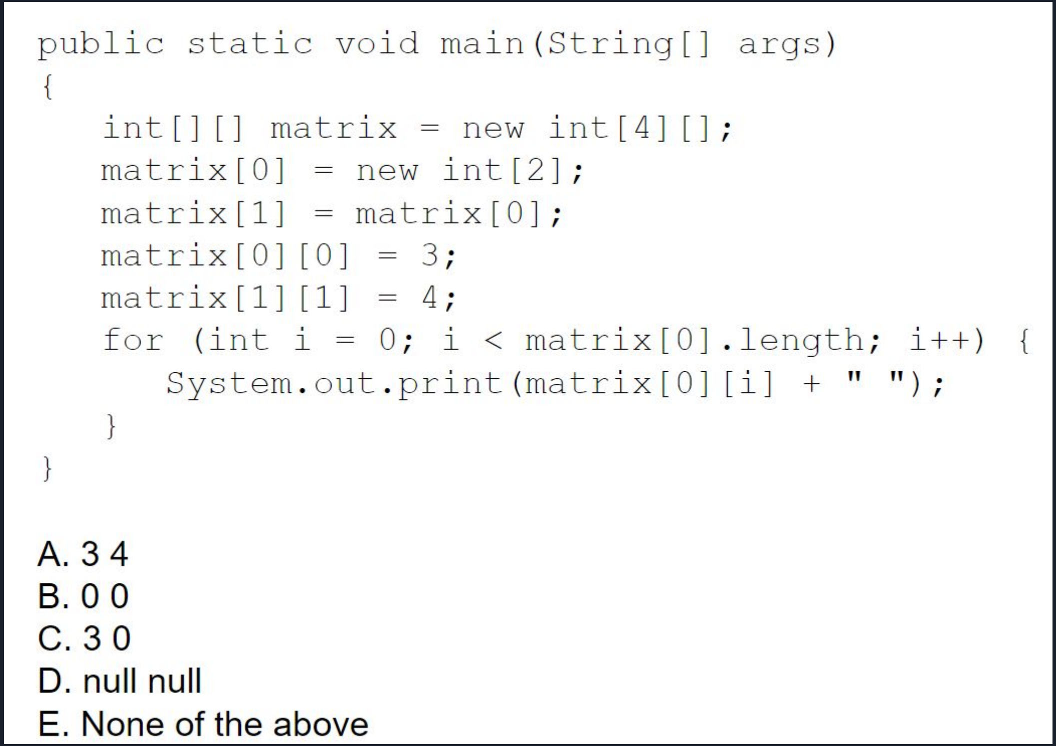

Q2. Which of the following would return null?

```
public class Matrix{
    public static void main(String[] args) {
        int[][] matrix = new int[5][3];
        matrix[0] = new int[] {1, 3, 4};
        matrix[1] = new int[5];
        matrix[2] = new int[] {12, 23, 11};
    }
}
```

a. matrix[3]
b. matrix[1][2]
c. matrix[4]
d. matrix[3] and matrix[4]
[✓ e. None of the above]

If we had initialized matrix with new int[5][], matrix[3] and matrix[4] would return null

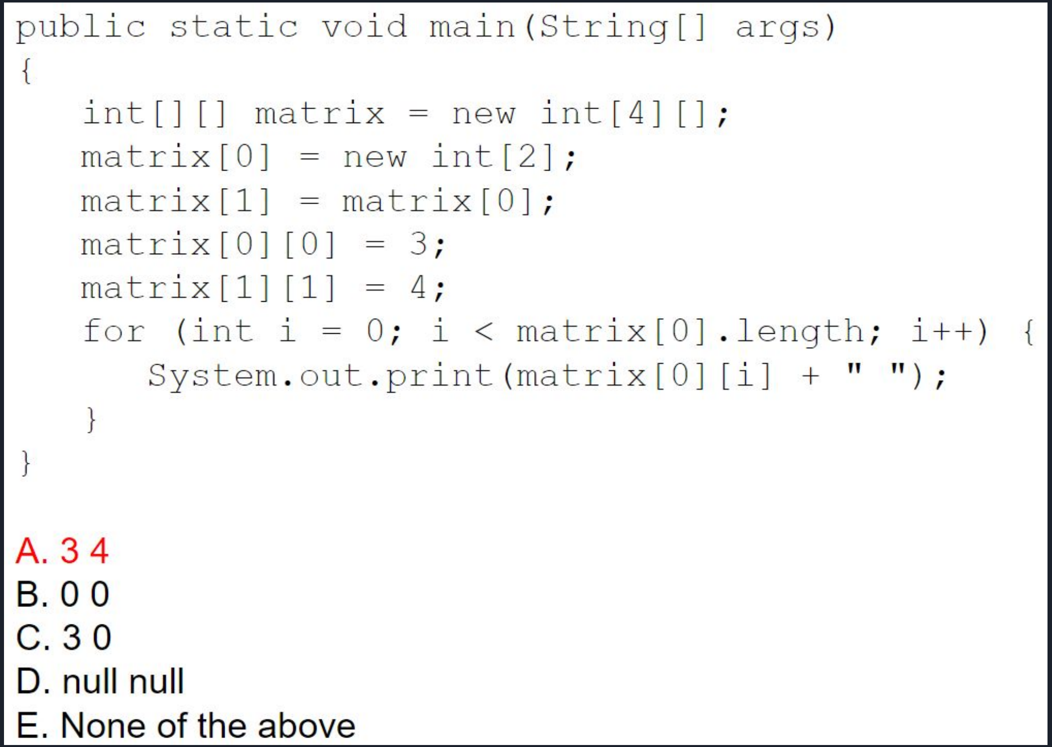
4b: Q7



```
public static void main(String[] args)
{
   int[][] matrix = new int[4][];
   matrix[0] = new int[2];
   matrix[1] = matrix[0];
   matrix[0][0] = 3;
   matrix[1][1] = 4;
   for (int i = 0; i < matrix[0].length; i++) {
      System.out.print(matrix[0][i] + " ");
   }
}
```

A. 3 4
B. 0 0
C. 3 0
D. null null
E. None of the above

4b:A7

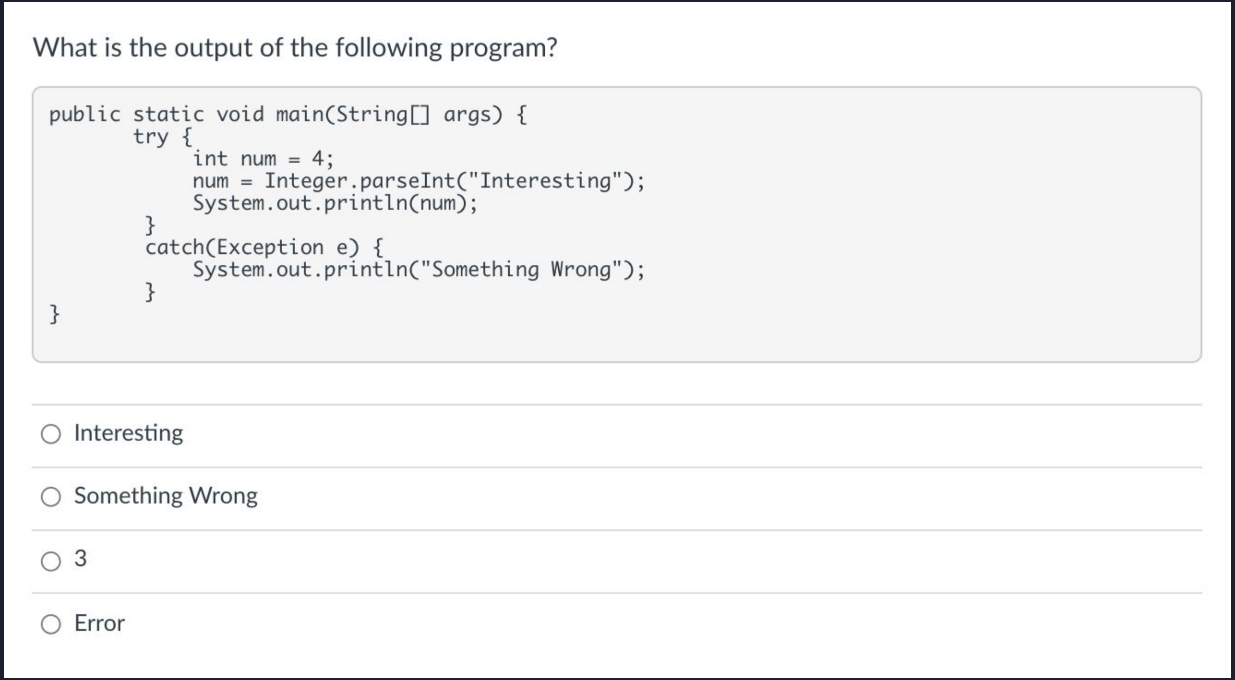


```
public static void main(String[] args)
{
   int[][] matrix = new int[4][];
   matrix[0] = new int[2];
   matrix[1] = matrix[0];
   matrix[0][0] = 3;
   matrix[1][1] = 4;
   for (int i = 0; i < matrix[0].length; i++) {
      System.out.print(matrix[0][i] + " ");
   }
}
```

A. 3 4
B. 0 0
C. 3 0
D. null null
E. None of the above

END OF MODULE 04B

5 Q1:

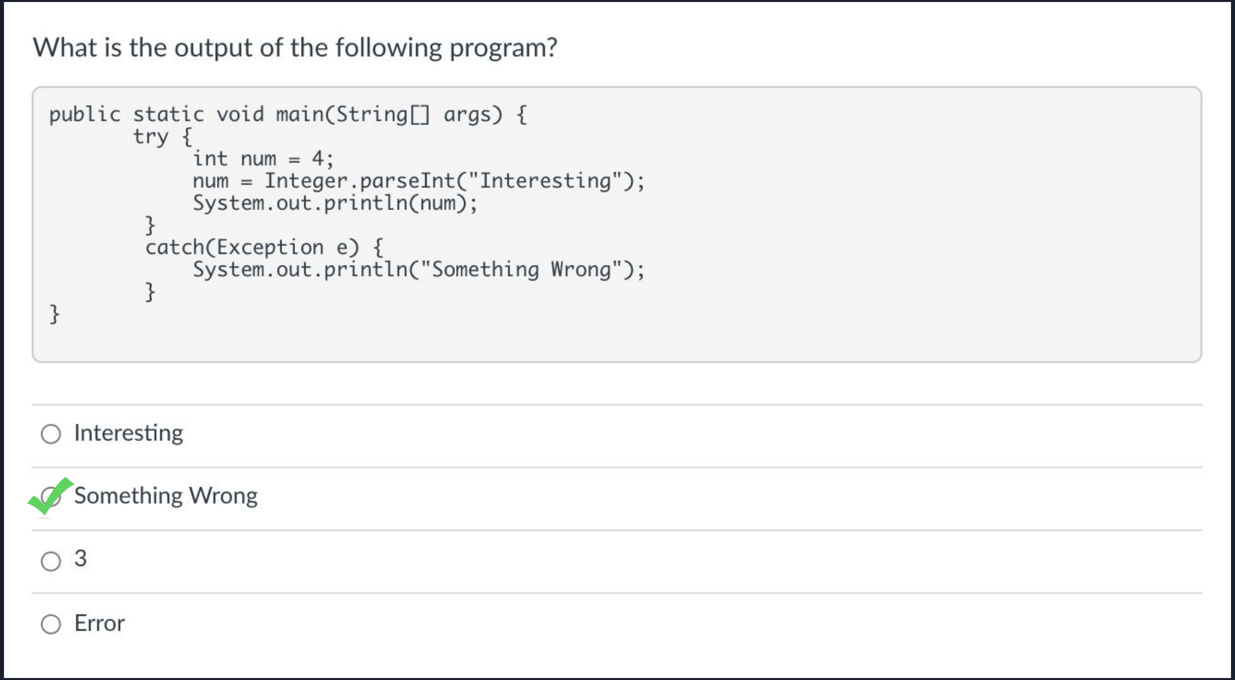


What is the output of the following program?

```
public static void main(String[] args) {
       try {
            int num = 4;
            num = Integer.parseInt("Interesting");
            System.out.println(num);
        }
        catch(Exception e) {
            System.out.println("Something Wrong");
        }
}
```

- ○ Interesting
- ○ Something Wrong
- ○ 3
- ○ Error

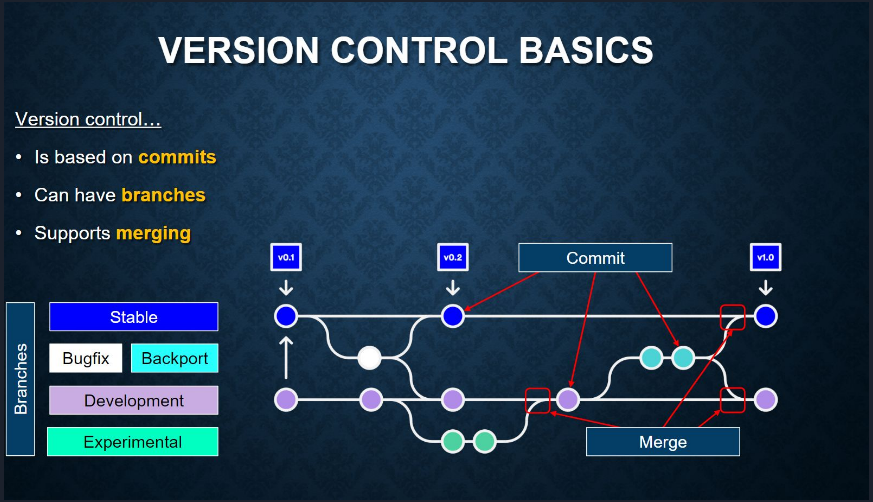
5 A1:


What is the output of the following program?

```
public static void main(String[] args) {
       try {
            int num = 4;
            num = Integer.parseInt("Interesting");
            System.out.println(num);
        }
        catch(Exception e) {
            System.out.println("Something Wrong");
        }
}
```

- ○ Interesting
- [✓ Something Wrong]
- ○ 3
- ○ Error

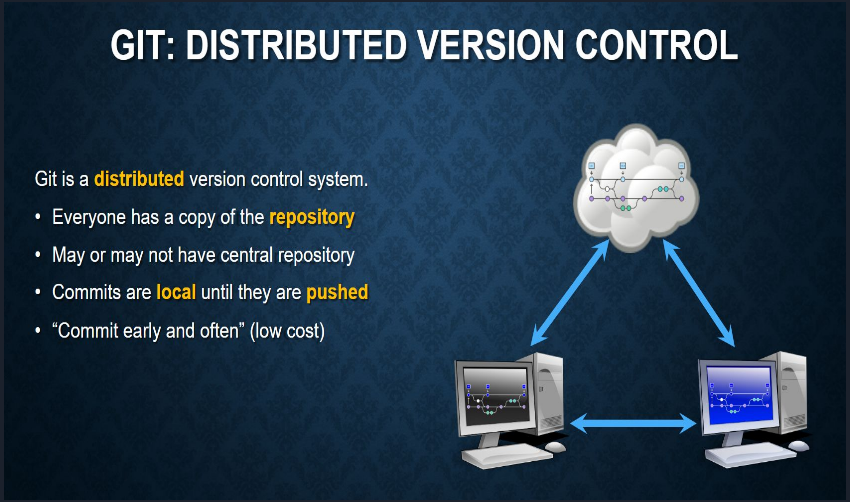
Review of Version Control



VERSION CONTROL BASICS

Version control…

- Is based on **commits**
- Can have **branches**
- Supports **merging**



GIT: DISTRIBUTED VERSION CONTROL

Git is a **distributed** version control system.

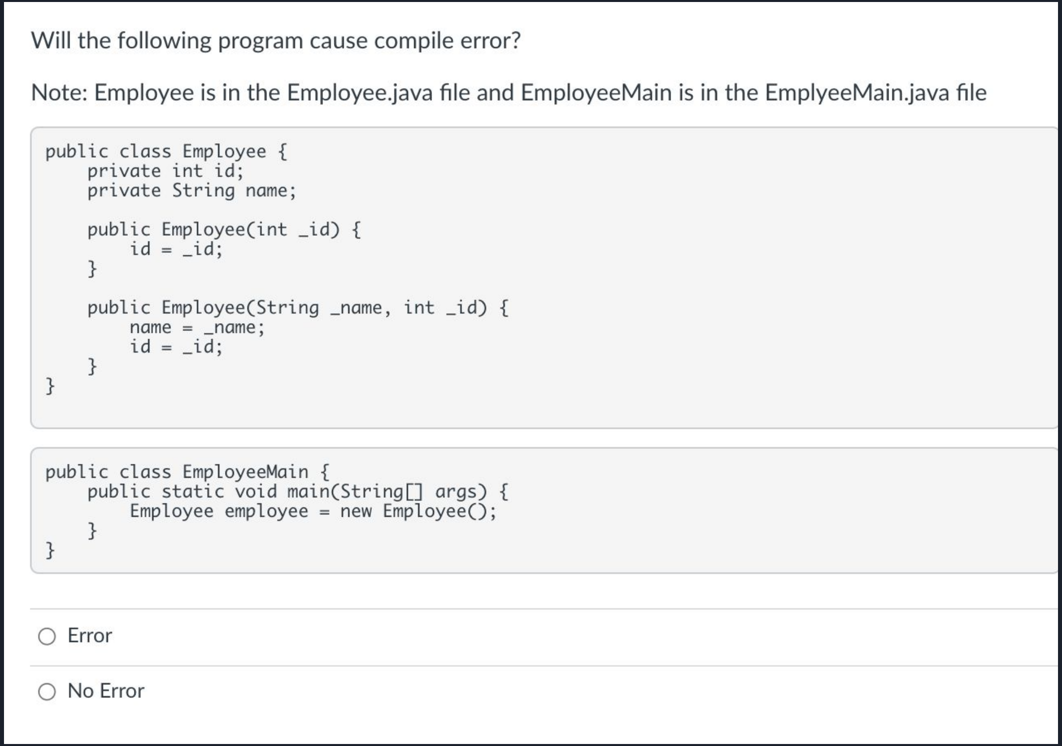
- Everyone has a copy of the **repository**
- May or may not have central repository
- Commits are **local** until they are **pushed**
- “Commit early and often” (low cost)

Review PT.2

- Clone: A clone operation makes a copy of a repository locally (does not capture future commits)
- Commit: Creates a changeset from one version to another (highlights changes)
- Push: A push operation pushes any changes up to a remote repository
- Pull: A pull operation pulls down any changes made to the remote repository

END OF MODULE 5

6 Q1:



Will the following program cause compile error?

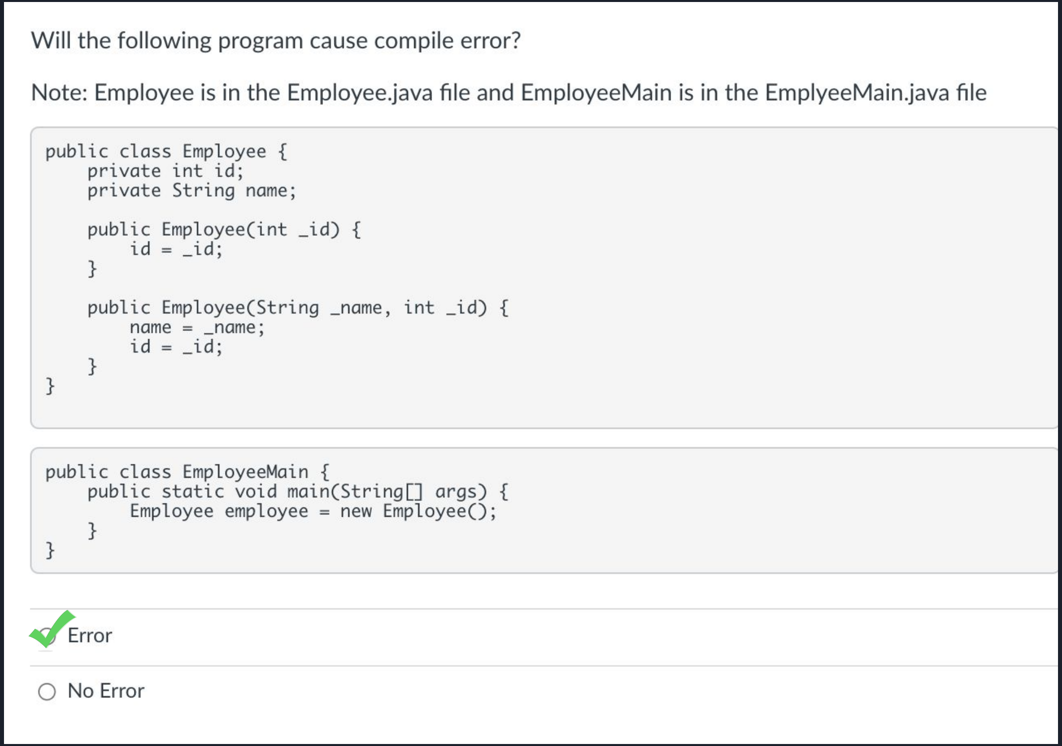
Note: Employee is in the Employee.java file and EmployeeMain is in the EmplyeeMain.java file

```
public class Employee {
    private int id;
    private String name;

    public Employee(int _id) {
        id = _id;
    }

    public Employee(String _name, int _id) {
        name = _name;
        id = _id;
    }
}
```

```
public class EmployeeMain {
    public static void main(String[] args) {
        Employee employee = new Employee();
    }
}
```

- ○ Error
- ○ No Error

6 A1:


Will the following program cause compile error?

Note: Employee is in the Employee.java file and EmployeeMain is in the EmplyeeMain.java file

```
public class Employee {
    private int id;
    private String name;

    public Employee(int _id) {
        id = _id;
    }

    public Employee(String _name, int _id) {
        name = _name;
        id = _id;
    }
}
```

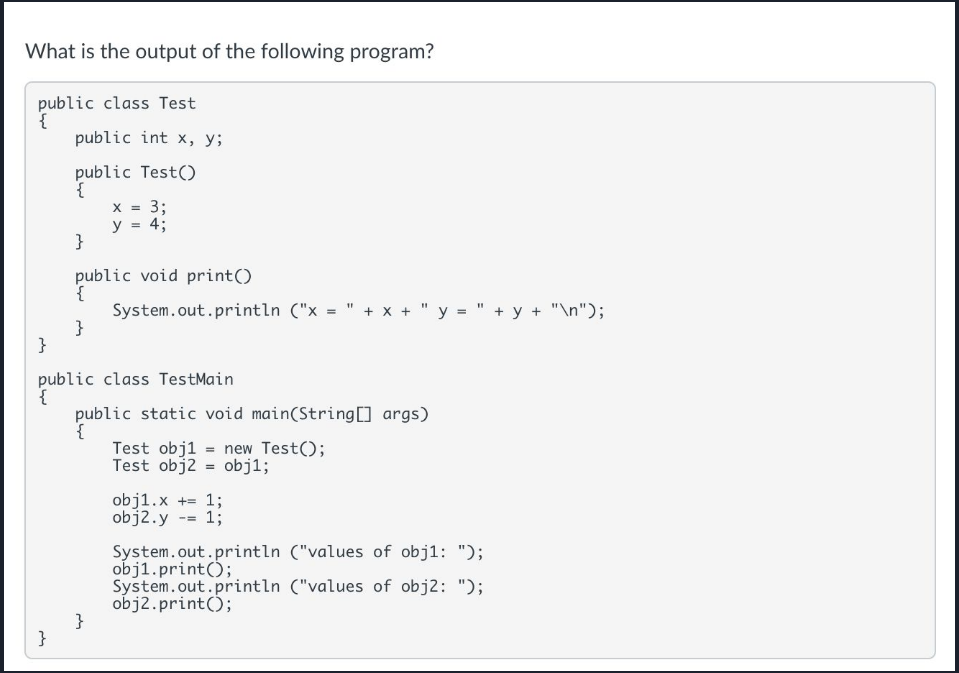
```
public class EmployeeMain {
    public static void main(String[] args) {
        Employee employee = new Employee();
    }
}
```

✓ Error

○ No Error

JAVA gives you default constructor automatically (which sets everything = default values) but if you (the programmer) make any constructor at all the default one that java gives u gets thrown away.

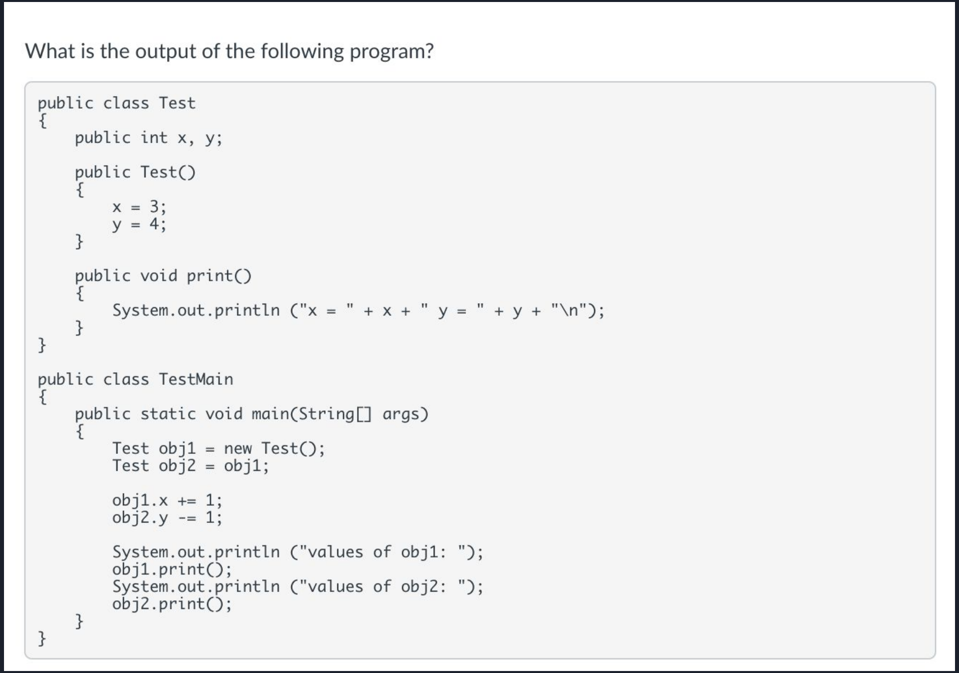
6 Q2:



What is the output of the following program?

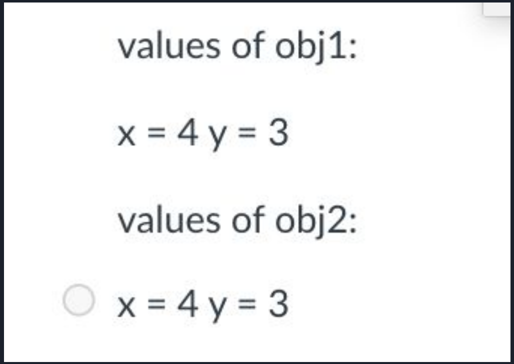
```
public class Test
{
    public int x, y;

    public Test()
    {
        x = 3;
        y = 4;
    }

    public void print()
    {
        System.out.println ("x = " + x + " y = " + y + "\n");
    }
}

public class TestMain
{
    public static void main(String[] args)
    {
        Test obj1 = new Test();
        Test obj2 = obj1;

        obj1.x += 1;
        obj2.y -= 1;

        System.out.println ("values of obj1: ");
        obj1.print();
        System.out.println ("values of obj2: ");
        obj2.print();
    }
}
```

6 A2:



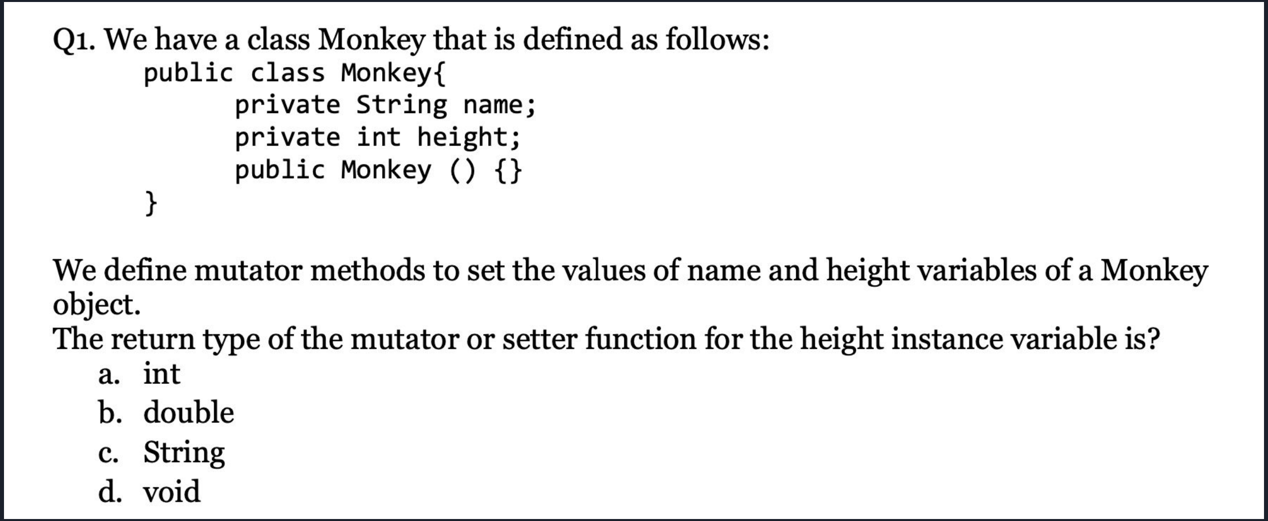
What is the output of the following program?

```
public class Test
{
    public int x, y;

    public Test()
    {
        x = 3;
        y = 4;
    }

    public void print()
    {
        System.out.println ("x = " + x + " y = " + y + "\n");
    }
}

public class TestMain
{
    public static void main(String[] args)
    {
        Test obj1 = new Test();
        Test obj2 = obj1;

        obj1.x += 1;
        obj2.y -= 1;

        System.out.println ("values of obj1: ");
        obj1.print();
        System.out.println ("values of obj2: ");
        obj2.print();
    }
}
```



```
values of obj1:

x = 4 y = 3

values of obj2:

x = 4 y = 3
```

6 Q3:



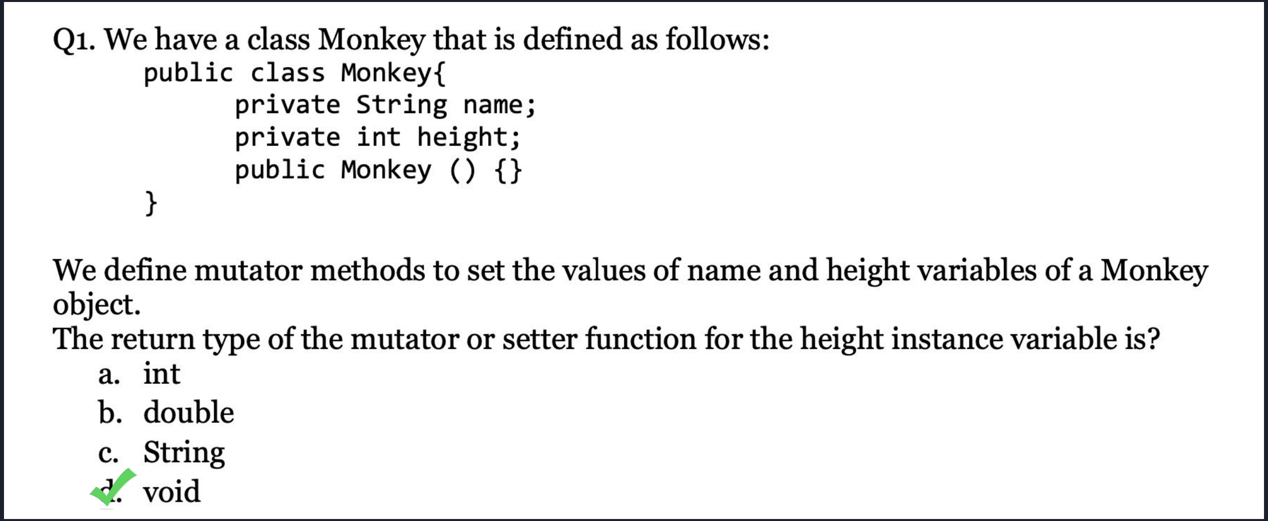
Q1. We have a class Monkey that is defined as follows:

```
public class Monkey{
      private String name;
      private int height;
      public Monkey () {}
}
```

We define mutator methods to set the values of name and height variables of a Monkey object.
The return type of the mutator or setter function for the height instance variable is?

a. int
b. double
c. String
d. void

6 A3:



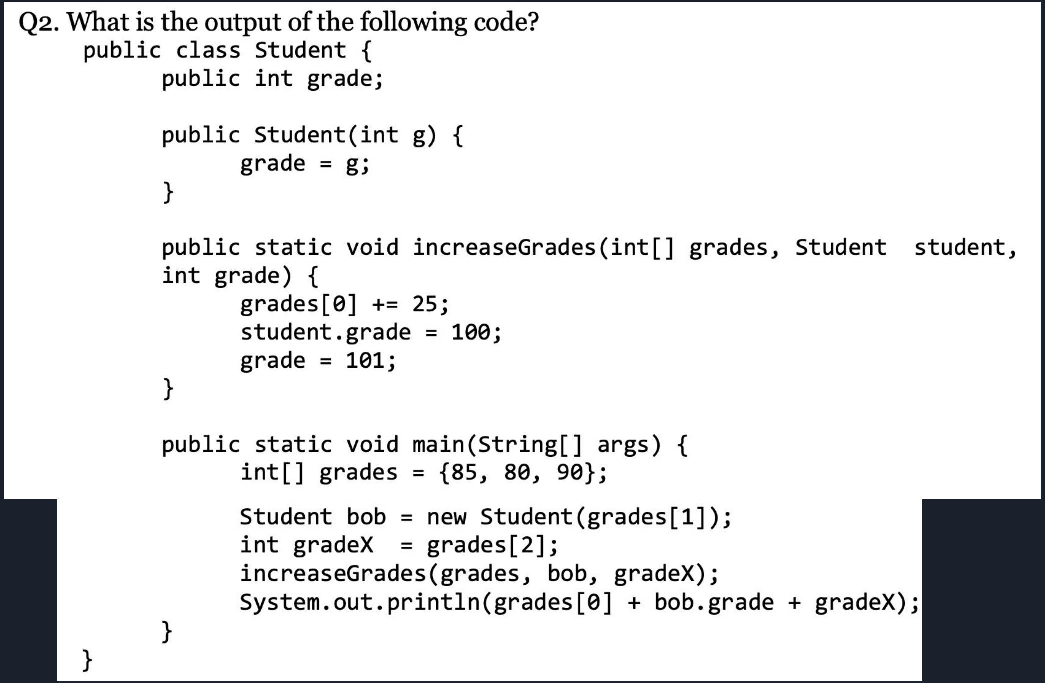
Q1. We have a class Monkey that is defined as follows:

```
public class Monkey{
      private String name;
      private int height;
      public Monkey () {}
}
```

We define mutator methods to set the values of name and height variables of a Monkey object.
The return type of the mutator or setter function for the height instance variable is?

a. int
b. double
c. String
[✓ d. void]

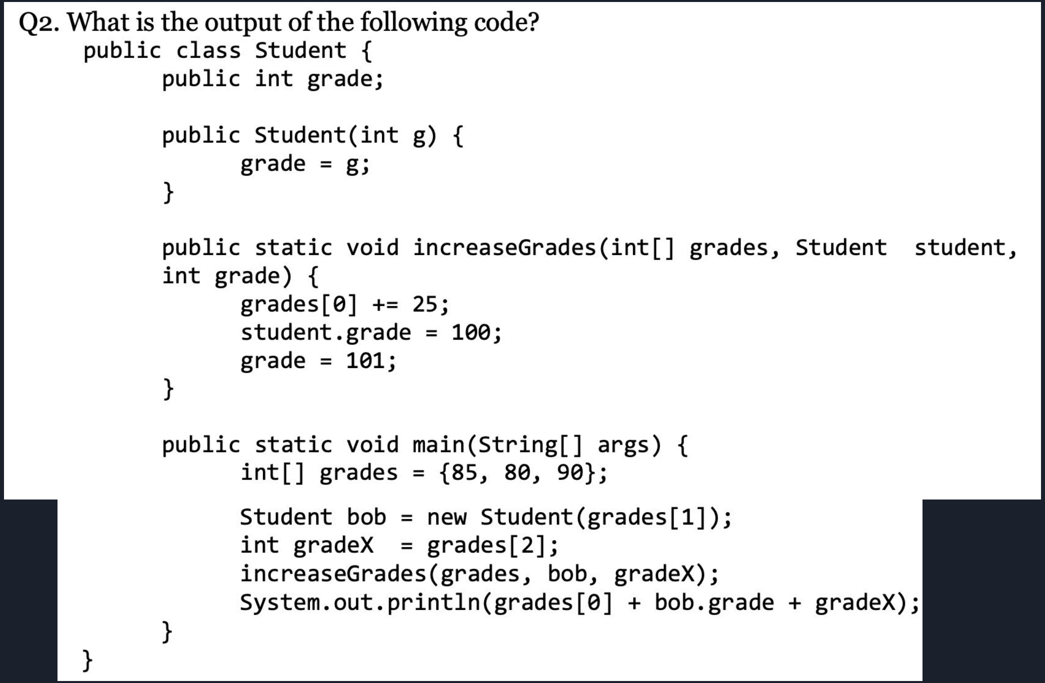
6 Q4:



Q2. What is the output of the following code?

```
public class Student {
      public int grade;

      public Student(int g) {
            grade = g;
      }

      public static void increaseGrades(int[] grades, Student  student,
      int grade) {
            grades[0] += 25;
            student.grade = 100;
            grade = 101;
      }

      public static void main(String[] args) {
            int[] grades = {85, 80, 90};
            Student bob = new Student(grades[1]);
            int gradeX  = grades[2];
            increaseGrades(grades, bob, gradeX);
            System.out.println(grades[0] + bob.grade + gradeX);
      }
}
```

6 A4:



Q2. What is the output of the following code?

```
public class Student {
    public int grade;

    public Student(int g) {
        grade = g;
    }

    public static void increaseGrades(int[] grades, Student  student,
    int grade) {
        grades[0] += 25;
        student.grade = 100;
        grade = 101;
    }

    public static void main(String[] args) {
        int[] grades = {85, 80, 90};
        Student bob = new Student(grades[1]);
        int gradeX  = grades[2];
        increaseGrades(grades, bob, gradeX);
        System.out.println(grades[0] + bob.grade + gradeX);
    }
}
```

300

6 Q5:

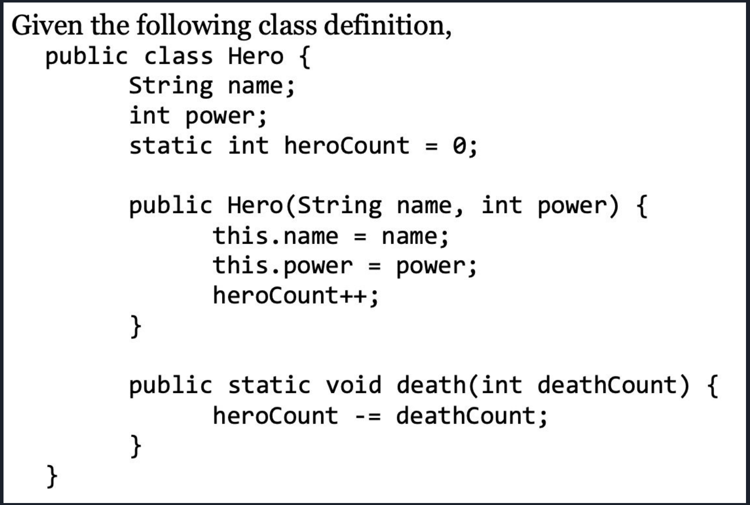
Given the following class definition,

```
public class Hero {
      String name;
      int power;
      static int heroCount = 0;

      public Hero(String name, int power) {
            this.name = name;
            this.power = power;
            heroCount++;
      }

      public static void death(int deathCount) {
            heroCount -= deathCount;
      }
}
```

What is the output of the following code snippet?
Assume that the main method is implemented and ran properly

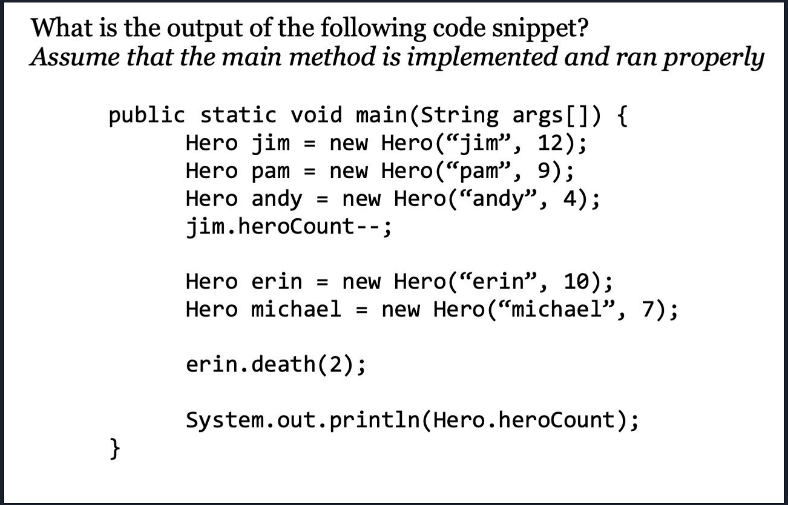
```
public static void main(String args[]) {
      Hero jim = new Hero(“jim”, 12);
      Hero pam = new Hero(“pam”, 9);
      Hero andy = new Hero(“andy”, 4);
      jim.heroCount--;

      Hero erin = new Hero(“erin”, 10);
      Hero michael = new Hero(“michael”, 7);

      erin.death(2);

      System.out.println(Hero.heroCount);
}
```

6 A5:



Given the following class definition,

```
public class Hero {
        String name;
        int power;
        static int heroCount = 0;

        public Hero(String name, int power) {
                this.name = name;
                this.power = power;
                heroCount++;
        }

        public static void death(int deathCount) {
                heroCount -= deathCount;
        }
}
```



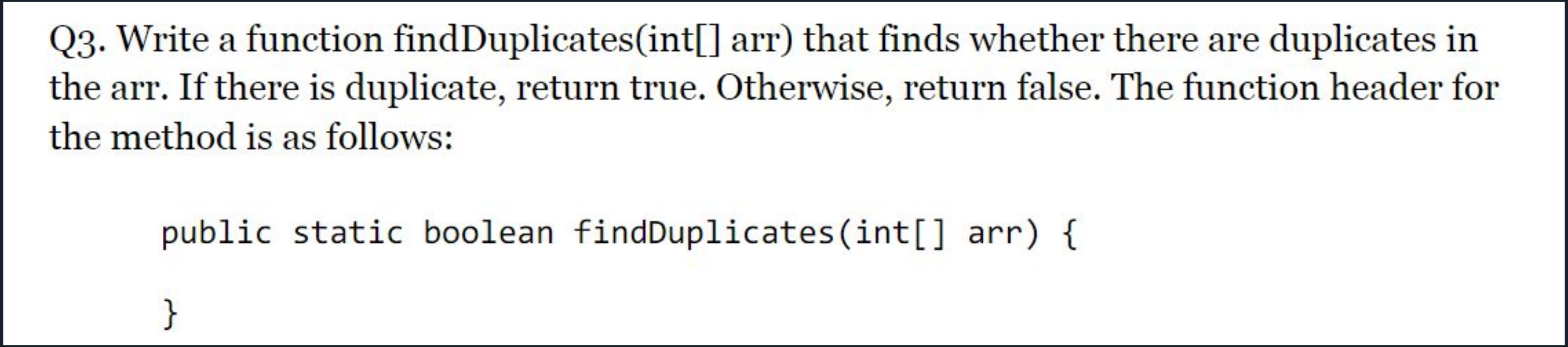
What is the output of the following code snippet?
Assume that the main method is implemented and ran properly

```
public static void main(String args[]) {
        Hero jim = new Hero(“jim”, 12);
        Hero pam = new Hero(“pam”, 9);
        Hero andy = new Hero(“andy”, 4);
        jim.heroCount--;

        Hero erin = new Hero(“erin”, 10);
        Hero michael = new Hero(“michael”, 7);

        erin.death(2);

        System.out.println(Hero.heroCount);
}
```

2

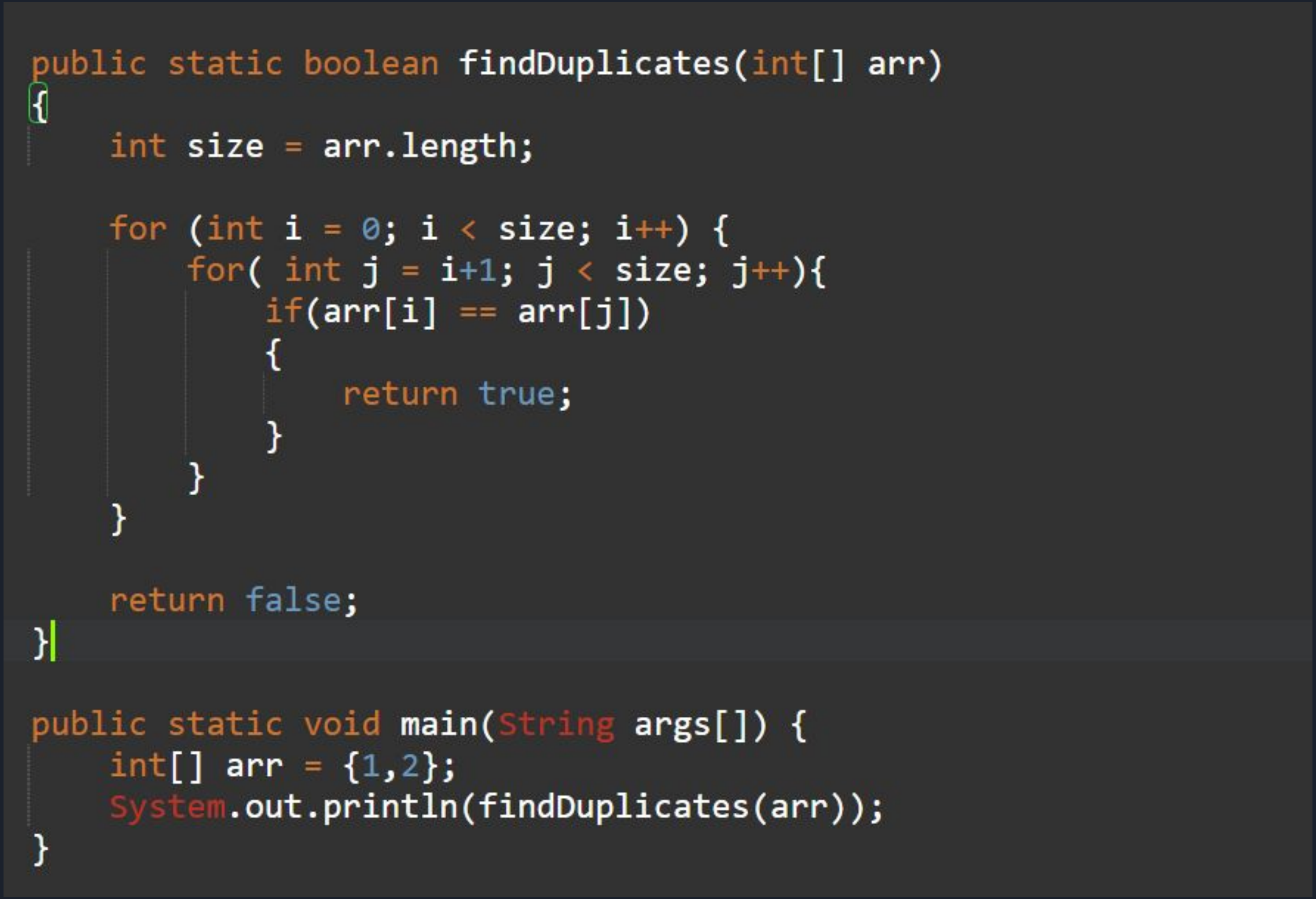
END OF MODULE 6

Coding Question



Q3. Write a function findDuplicates(int[] arr) that finds whether there are duplicates in the arr. If there is duplicate, return true. Otherwise, return false. The function header for the method is as follows:

```
public static boolean findDuplicates(int[] arr) {

}
```



```java
public static boolean findDuplicates(int[] arr)
{
    int size = arr.length;

    for (int i = 0; i < size; i++) {
        for( int j = i+1; j < size; j++){
            if(arr[i] == arr[j])
            {
                return true;
            }
        }
    }

    return false;
}

public static void main(String args[]) {
    int[] arr = {1,2};
    System.out.println(findDuplicates(arr));
}
```