Back to Results Table 1 of 8

1. Polymorphism allows you to write code that's more _____.

Show Results 19/21 Students Answered

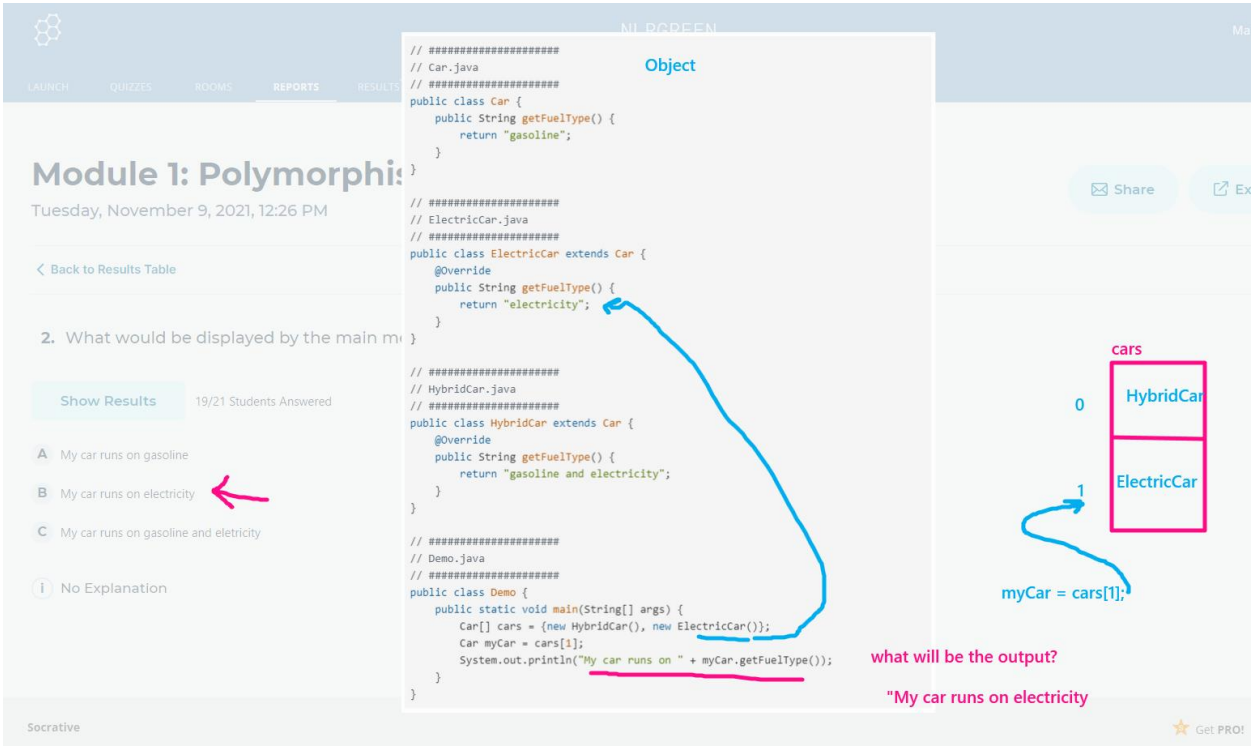
A specific

B generic

i No Explanation

Parent class specific
FarmAnimal animal = new Cow();
generic specific
animal = new Goat();

animal is a FarmAnimal object so it can hold any of the children of FarmAnimal through Polymorphism



5. When an interface method has no access modifier, it is _____

Show Results 19/21 Students Answered

A public

B private

C protected

D static

i No Explanation

public interface Singable{
public String getName();
String getSound();

By definition all interface methods are always abstract and public

ember 9, 2021, 12:26 PM

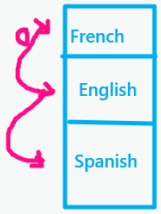
Table

uld be displayed by the main method
ntion to spelling, capitalization, and sp

ers Show Names 18/21 Students Ar

la!
la!
la!
la!
la!
la!
la!
la!
la!
ch Bonjour! Spanish Hola!
la!
la!
la

```java
// ####################
// Greeting.java
// ####################

public interface Greeting {
     String getGreeting();
}

// ####################
// English.java
// ####################

public class English implements Greeting {
    public String getGreeting() {
        return "Hello!";
    }
}

// ####################
// French.java
// ####################

public class French implements Greeting {
    public String getGreeting() {
        return "Bonjour!";
    }
}

// ####################
// Spanish.java
// ####################

public class Spanish implements Greeting {
    public String getGreeting() {
        return "Hola!";
    }
}

// ####################
// Demo.java
// ####################

public class Demo {

    public static void main(String[] args) {
        Greeting[] greetings =
            new Greeting[] { new French(), new English(), new Spanish() };

        for(Greeting g : greetings ) {
            System.out.print( g.getGreeting() + " " );
        }
    }
}
```



Bonjour! Hello! Hola!

for Each loop looping through the array called greetings

Module 1: Polymorphism (NLR5)

Tuesday, November 9, 2021, 12:26 PM

Back to Results Table

7. Select the statement below that's true.

Show Results 18/21 Students Answered

A Shape.java will not compile successfully

B Rectangle.java will not compile successfully

C Both files compile successfully

i No Explanation

```java
// ################################################
// Shape.java
// ################################################

public interface Shape {
    double getArea();
    double getPerimeter();
}

// ################################################
// Rectangle.java
// ################################################

public class Rectangle implements Shape {
    private double length;
    private double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    public double getArea() {
        return length * width;
    }
}
```

public abstract

where is getPerimeter()????

8. Select all of the following that are examples of interfaces:

FarmAnimal

Show Results 18/21 Students Answered

A ArrayList

B Map

C String

D List

E HashMap

i No Explanation

programming to the interface - so we can have polymorphic behavior

List<String> myList = new ArrayList<>();

Generic

Generic

Specific

Map<String, Integer> myMap = new HashMap<>();

```
…
System.out.println("Movie ticket prices: ");
System.out.println("1. Adult - $14.00");
System.out.println("2. Child - $8.00");
 System.out.println("3. Senior - $11.00");
System.out.print("Enter choice: ");
int choice = Integer.parseInt(input.nextLine());
       if (choice == 1) {
          total = quantity * 14;
       } else if (choice == 2) {
           total = quantity * 8;
       }  else if (choice == 3) {
            total = quantity * 11;
       }  else {
            System.out.println("Invalid entry");
       }

  …
```

You CANNOT use a switch statement for relational > >=

Can only use it for == .equals

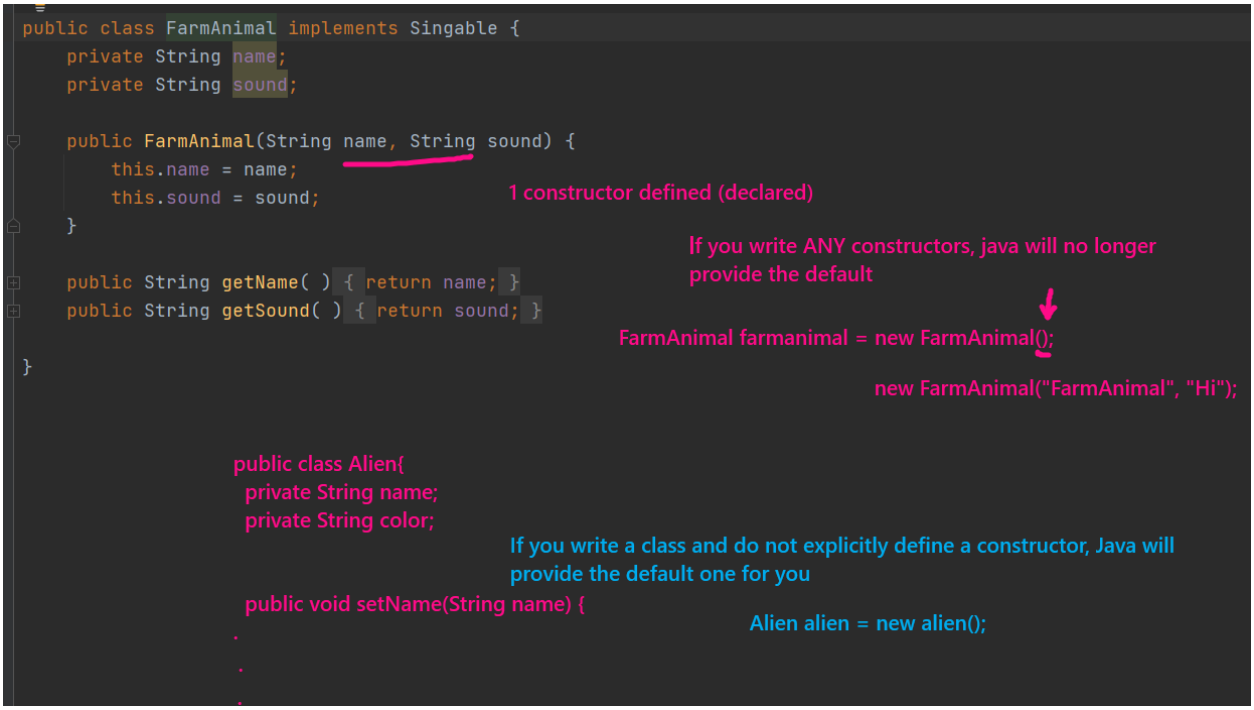
```
…
System.out.println("Movie ticket prices: ");
System.out.println("1. Adult - $14.00");
System.out.println("2. Child - $8.00");
 System.out.println("3. Senior - $11.00");
System.out.print("Enter choice: ");
int choice = Integer.parseInt(input.nextLine());
switch (choice) {
       case 1:
            total = quantity * 14;
            break;
       case 2:
            total = quantity * 8;
            break;
       case 3:
            total = quantity * 11;
            break;
       default:
            System.out.println("Invalid entry");
  }
```

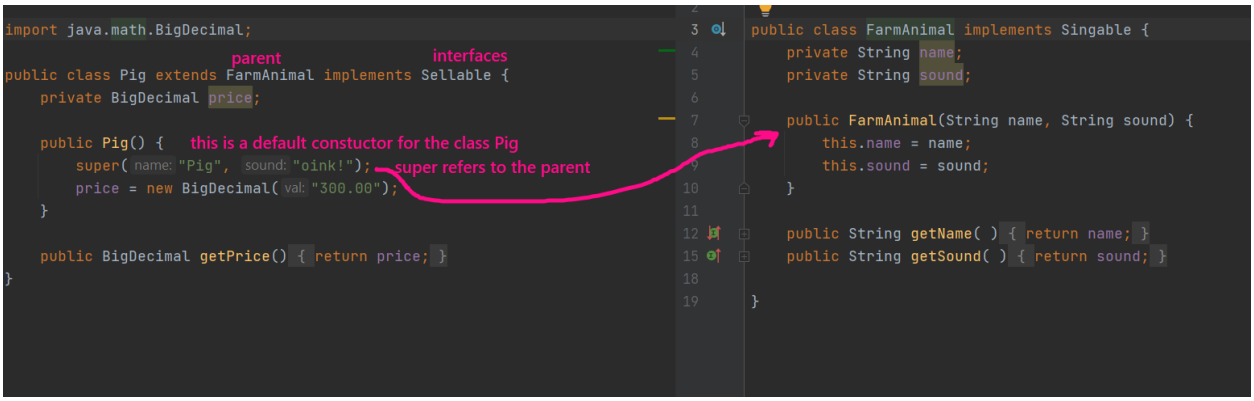
```
…
System.out.println("Movie ticket prices: ");
System.out.println("1. Adult - $14.00");
System.out.println("2. Child - $8.00");
 System.out.println("3. Senior - $11.00");
System.out.print("Enter choice: ");
int choice = Integer.parseInt(input.nextLine());
       if (choice == 1) {
          total = quantity * 14;
       } else if (choice == 2) {
           total = quantity * 8;
       }  else if (choice == 3) {
            total = quantity * 11;
       }  else {
            System.out.println("Invalid entry");
       }

  …
```

break takes you out of the block of code you are in

```
…
System.out.println("Movie ticket prices: ");
System.out.println("1. Adult - $14.00");
System.out.println("2. Child - $8.00");
 System.out.println("3. Senior - $11.00");
System.out.print("Enter choice: ");
int choice = Integer.parseInt(input.nextLine());
switch (choice) {
       case 1:
            total = quantity * 14;
            break;
       case 2:
            total = quantity * 8;
            break;
       case 3:
            total = quantity * 11;
            break;
       default:
            System.out.println("Invalid entry");
  }
```


private String name;
private String sound;
public FarmAnimal(String name, String sound) {
this.name = name;
this.sound = sound;
}
public String getName( ) { return name; }
public String getSound( ) { return sound; }
1 constructor defined (declared)
FarmAnimal farmanimal = new FarmAnimal();
new FarmAnimal("FarmAnimal", "Hi");
public class Alien{
private String name;
private String color;
public void setName(String name) {
If you write a class and do not explicitly define a constructor, Java will provide the default one for you
Alien alien = new alien();


public class FarmAnimal implements Singable {
private String name;
private String sound;
public FarmAnimal(String name, String sound) {
this.name = name;
this.sound = sound;
}
public String getName( ) { return name; }
public String getSound( ) { return sound; }
1 constructor defined (declared)
If you write ANY constructors, java will no longer provide the default
FarmAnimal farmanimal = new FarmAnimal();
new FarmAnimal("FarmAnimal", "Hi");
public class Alien{
private String name;
private String color;
public void setName(String name) {
If you write a class and do not explicitly define a constructor, Java will provide the default one for you
Alien alien = new alien();


parent
interfaces
this is a default constuctor for the class Pig
super refers to the parent