




1. Which choice below best represents a method signature?

Show Results


19/20 Students Answered

- ☐ A The signature of a method is the name of the method and a summary of the method description. 
- ☐ B The signature of a method is the name of the method and the names of its parameters. `myMethod(var1, var2){`
- ☐ C The signature of a method is the name of the method and any side effects it may have. 
- ☐ D The signature of a method is the name of the method and the types of its parameters. `myMethod(int var1, double var2)` 
- ☐ E I don't know

4. Polymorphism allows us to write code that operates in a _____ manner.

Show Results

19/20 Students Answered

- ☐ A specific
- ☒ B generic 
- ☐ C I don't know

Array of animals and then can use `animal.getSound();` which is more generic

```
6. // #####
// 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!";
    }
}

// #####
// GreetingDemo.java
// #####

public class GreetingDemo {
    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() + " ";
        }
    }
}
```

 abstract method that any class that implements Greeting MUST provide the body for (override the method)

Array greetings
0 French object
1 English object
2 Spanish object

Output:

Bonjour! Hello! Hola!

What is the output displayed by the main method of GreetingDemo when it is executed? (Pay attention to spaces!)

```

7. //
#####
// Shape.java
//
#####

```

```

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

```

Contract -- anyone that implements this interface MUST override both of the abstract methods

```

//
#####
// 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;
    }
}

```

constructor

? ☹

Rectangle will not compile

Select the statement(s) below that are true. (Choose all that apply)

```

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;
    }
}

```

rectangle.length
rectangle.width

3
5

ShapeDemo

main method:

Rectangle rectangle = new Rectangle(3, 5);

double area = rectangle.getArea();

Select the statement(s) below that are true. (Choose all that apply)

Show Results

18/20 Students Answered

- ☐ A Shape.java will not compile successfully
- ☐ B Rectangle.java will not compile successfully
- ☐ C Both files will compile successfully

```

public class Triangle implements Shape{
    private double base;
    private double height;

```

```

    public Triangle(double base, double height){
        this.base = base;
        this.height = height;
    }
}

```

```

    public double getArea(){
        return base * height * 0.5;
    }
}

```

Use of this – implementing in a method.

Static key word

Super

Polymorphism

Abstract