

Which two will compile, and can be run successfully using the following command?

★ 1 punto

java Fred1 hello walls

- ☐ class Fred 1{ public static void main(String args) { System.out.println(args[1]); }}
- ☐ class Fred1 { public static void main(String[] args) { System.out.println(args[1]); }}
- ☒ class Fred1 { public static void main(String[] args) { System.out.println(args); }}
- ☐ abstract class Fred1 { public static void main(String[] args) { System.out.println(args[2]); }}

- la primera opción tiene un espacio en el nombre de la clase y faltan corchetes, no es un método main
- la segunda opción si compila e imprime walls
- la tercera opción imprime un array con su hashCode
- la clase abstracta sí compila, pero falla porque le da un argumento en el index 2 pero se sale del tamaño del array

```
class Foo{
    public void addFive(){ a += 5; System.out.println("f"); }
}
class Bar extends Foo{
    public int a = 8;
    public void addFive(){ this.a += 5; System.out.println("b")
}
Invoked with:
    Foo f = new Bar();
    f.addFive();
    System.out.println(f.a);
```

- ☐ f 8
- ☐ b 13
- ☐ An exception is thrown at runtime.
- ☐ f 13
- ☐ f 3
- ☐ Compilation fails
- ☐ b 8
- ☐ b 3

miguel no sigue con esta pregunta, pero la respuesta es: **falla la compilación**. Parece ser porque la variable a no está inicializada.

What is the result? *

1 punto

```
class MySort implements Comparator<Integer>{  
    public int compare(Integer x, Integer y){  
        return y.compareTo(x);  
    }  
}
```

And the code fragment:

```
Integer[] primes = {2,7,5,3};  
MySort ms = new MySort();  
Arrays.sort(primes, ms);  
for(Integer p2:primes){System.out.print(p2+"");}
```

- ☐ 2 7 5 3
- ☐ 7 5 3 2
- ☐ Compilation fails
- ☐ 2 3 5 7

la respuesta es 7 5 3 2 ya que la comparación (método compareTo) empieza con el parámetro y. Por lo que el orden será de forma descendente