

```

1 package edu.val.cle.atsistemas.java8;
2
3 import java.util.function.Predicate;
4
5 public class Panda {
6
7     int age;
8
9     public static void main(String[] args) {
10         Panda p1 = new Panda();
11         p1.age = 1;
12         check(p1, p->p.age<5);
13
14     }
15
16     private static void check (Panda panda, Predicate<Panda> p)
17     {
18         String resultado = p.test(panda) ? "match" : "no match";
19         System.out.println(resultado);
20     }
21 }
22
23 /**
24  * OPCIONES
25  *
26  * a) match
27  * b) not match
28  * c) error compilación en línea 12
29  * d) error compilación en línea 16
30  * e) error compilación en línea 18
31  * f) lanza una RuntimeException*/
32

```

```

1 package edu.val.cle.atsistemas.java8;
2
3 public class InterfacesFuncionales {
4
5     public interface Transport {
6         public int go();
7         public boolean equals (Object o);
8     }
9
10    public abstract class Car {
11        public abstract Object arranca (int duracion);
12    }
13
14    public interface Train extends Transport {}
15
16    public interface Locomotive extends Train {
17        public int getSpeed();
18    }
19
20    public interface Spaceship extends Transport {
21        default int maxPasajeros() { return 10;}
22    }
23
24    public interface Boat {
25        int hashCode();
26        int hashCode(String input);
27    }
28 }
29 /**
30  * Qué interfaces son funcionales?
31  * a) Boat b) Car c) Locomotive d) SpaceShip e) Transport
32  * f)Train g) Ninguna*/

```

```

1 package edu.val.cle.atsistemas.java8;
2
3 public interface Secret {
4
5     String magic (double d);
6
7 }
8
9 class MySecret implements Secret {
10
11     @Override
12     public String magic(double d) {
13         return "Poof";
14     }
15 }
16
17 /**
18  * Qué lambdas pueden sustituir a MySecret
19  *
20  * a) (e)->"Poof"
21  * b) (e)-> {"Poof"}
22  * c) (e)-> {String e ="";"Poof"}
23  * d) (e)-> {String e ="";return "Poof";}
24  * e) (e)-> {String e ="";return "Poof"}
25  * f) (e)-> {String f ="";return "Poof";}
26  *
27  */
28

```

```
1 package edu.val.cle.atsistemas.java8;
2
3 import java.util.function.BinaryOperator;
4 import java.util.function.Consumer;
5
6 public class PruebaIF {
7
8     public void method ()
9     {
10         x ((var x) -> {}, (var x, var y) -> false);
11     }
12
13     public void x (Consumer<String> x, BinaryOperator<Boolean> y)
14     {}
15
16 }
17
18 /**
19  * Marque las correctas:
20  * a) No compila porque una de las variables se llama x
21  * b) No compila porque una de las variables se llama y
22  * c) No compila por otra razon
23  * d) Compila y la x es del mismo tipo en cada lambda
24  * e) Compila y la x es de distinto tipo en cada lamda
25  */
26
27
```

```
1 package edu.val.cle.atsistemas.java8;
2
3 import java.util.function.Function;
4
5 public class TestFunction {
6
7     public static void main(String[] args) {
8
9         Function<Integer, Integer> s = a->a+4;
10        Function<Integer, Integer> t = a->a*3;
11        Function<Integer, Integer> c = s.compose(t);
12        System.out.println(c.apply(1));
13    }
14
15 }
16
17 /**
18  * Qué devuelve este código
19  *
20  * a) 7
21  * b) 15
22  * c) No compila por los tipos de las expresiones lambda
23  * d) No compila por la llamada a compose
24  * e) No compila por otra razón
25  *
26  */
27
```