```java
package edu.val.cle.atsistemas.java8;

import java.util.function.Predicate;

public class Panda {

    int age;

    public static void main(String[] args) {
            Panda p1 = new Panda();
            p1.age = 1;
            check(p1, p->p.age<5);

    }

    private static void check (Panda panda, Predicate<Panda> p)
    {
        String resultado = p.test(panda) ? "match" : "no match";
        System.out.println(resultado);
    }
}

/**
 * OPCIONES
 *
 * a) match
 * b) not match
 * c) error compilación en linea 12
 * d) error compilación en linea 16
 * e) error compilación en linea 18
 * f) lanza una RuntimeException*/

```

```java
package edu.val.cle.atsistemas.java8;

public class InterfacesFuncionales {

    public interface Transport {
        public int go();
        public boolean equals (Object o);
    }

    public abstract class Car {
        public abstract Object arranca (int duracion);
    }

    public interface Train extends Transport {}

    public interface Locomotive extends Train {
        public int getSpeed();
    }

    public interface Spaceship extends Transport {
        default int maxPasajeros() { return 10;}
    }

    public interface Boat {
        int hashCode();
        int hashCode(String input);
    }
}
/**
 * Qué interfaces son funcionales?
 * a) Boat b) Car c) Locomotive d) SpaceShip e) Transport
 * f)Train g) Ninguna*/
```

```java
package edu.val.cle.atsistemas.java8;

public interface Secret {

    String magic (double d);

}

class MySecret implements Secret {

    @Override
    public String magic(double d) {
        return "Poof";
    }
}

/**
 * Qué lambdas pueden sustituir a MySecret
 *
 * a) (e)->"Poof"
 * b) (e)-> {"Poof"}
 * c) (e)-> {String e ="";"Poof"}
 * d) (e)-> {String e ="";return "Poof";}
 * e) (e)-> {String e ="";return "Poof"}
 * f) (e)-> {String f ="";return "Poof";}
 *
 */
```

```java
package edu.val.cle.atsistemas.java8;

import java.util.function.BinaryOperator;
import java.util.function.Consumer;

public class PruebaIF {

    public void method ()
    {
        x ((var x) -> {}, (var x, var y) -> false);
    }

    public void x (Consumer<String> x, BinaryOperator<Boolean> y)
    {}

}

/**
 * Marque las correctas:
 * a) No compila porque una de las variables se llama x
 * b) No compila porque una de las variables se llama y
 * c) No compila por otra razon
 * d) Compila y la x es del mismo tipo en cada lambda
 * e) Compila y la x es de distinto tipo en cada lamda
 */

```

```java
package edu.val.cle.atsistemas.java8;

import java.util.function.Function;

public class TestFunction {

    public static void main(String[] args) {

        Function<Integer, Integer> s = a->a+4;
        Function<Integer, Integer> t = a->a*3;
        Function<Integer, Integer> c = s.compose(t);
        System.out.println(c.apply(1));
    }

}

/**
 * Qué devuelve este código
 *
 * a) 7
 * b) 15
 * c) No compila por los tipos de las expresiones lambda
 * d) No compila por la llamada a compose
 * e) No compila por otra razón
 *
 * */

```