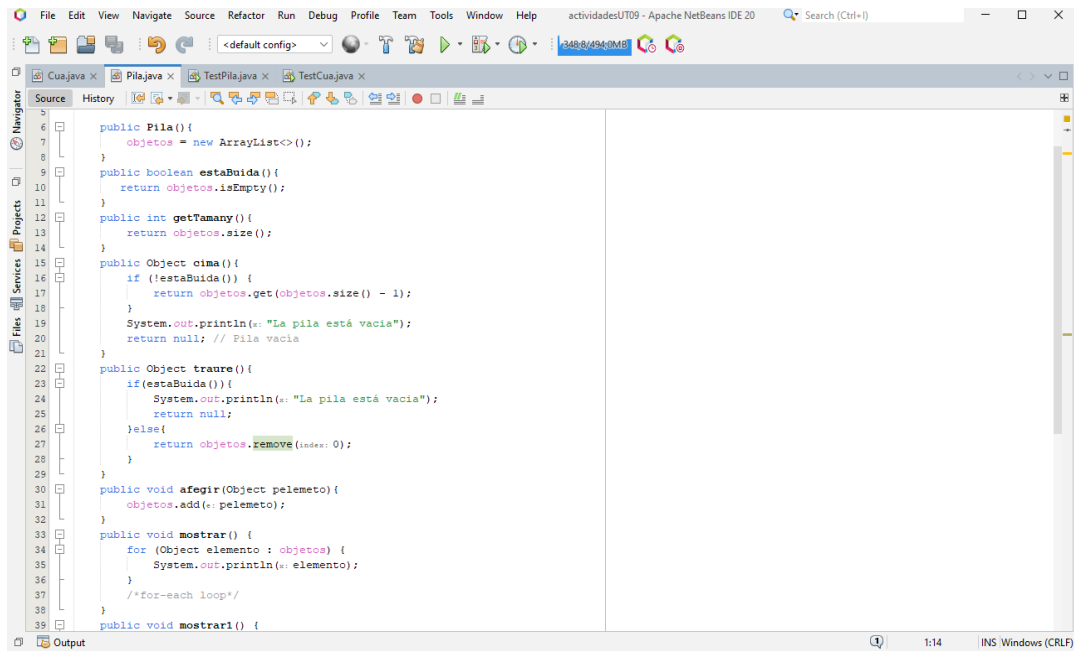


The screenshot shows the Apache NetBeans IDE with the 'TestCua.java' file open. The code defines a 'Cua' class with an 'inicio()' method that initializes a 'Cua' object and calls 'afegir()' with various parameters. A 'main()' method creates a 'TestCua' object and calls 'inicio()'. The 'Output' window at the bottom shows the execution results: 'run:', 'La cua està buida', 'Ramón', '17', 'true', '19.445', 'false', '17', 'true', '19.445', 'false', and 'Ramón Moreno'.

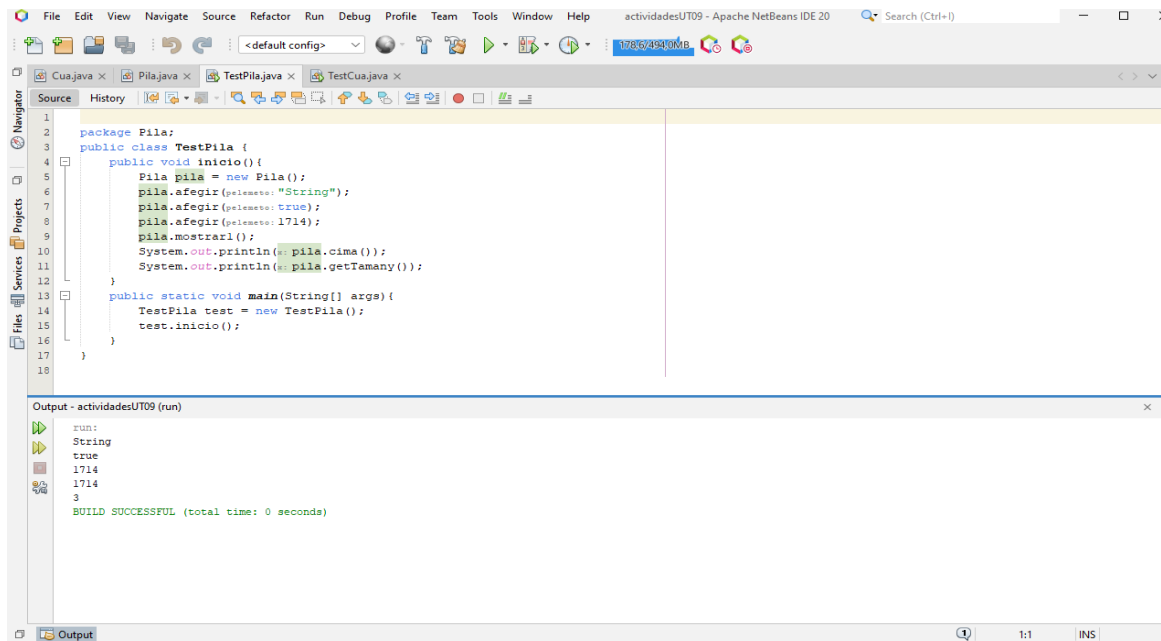
## Clase Pila:



The screenshot shows the NetBeans IDE with the 'Pila.java' file open. The code defines a stack class with methods for initialization, checking emptiness, getting size, popping, pushing, and displaying elements.

```
5 public Pila() {
6     objetos = new ArrayList<>();
7 }
8
9 public boolean estaBuida() {
10     return objetos.isEmpty();
11 }
12
13 public int getTamany() {
14     return objetos.size();
15 }
16
17 public Object cima() {
18     if (!estaBuida()) {
19         return objetos.get(objetos.size() - 1);
20     }
21     System.out.println("La pila está vacía");
22     return null; // Pila vacía
23 }
24
25 public Object traure() {
26     if (estaBuida()) {
27         System.out.println("La pila está vacía");
28         return null;
29     } else {
30         return objetos.remove(index: 0);
31     }
32 }
33
34 public void afegir(Object pelemeto) {
35     objetos.add(e: pelemeto);
36 }
37
38 public void mostrar() {
39     for (Object elemento : objetos) {
40         System.out.println(e: elemento);
41     }
42     /*for-each loop*/
43 }
44
45 public void mostrar1() {
```

## TestPila:



The screenshot shows the NetBeans IDE with the 'TestPila.java' file open. The code creates a TestPila class with a main method that tests the Pila class. Below the code, the output window shows the results of the test.

```
1 package Pila;
2 public class TestPila {
3     public void inicio() {
4         Pila pila = new Pila();
5         pila.afegir(pelemeto: "String");
6         pila.afegir(pelemeto: true);
7         pila.afegir(pelemeto: 1714);
8         pila.mostrar1();
9         System.out.println(e: pila.cima());
10        System.out.println(e: pila.getTamany());
11    }
12
13    public static void main(String[] args) {
14        TestPila test = new TestPila();
15        test.inicio();
16    }
17 }
18
```

Output - actividadesUT09 (run)

```
run:
String
true
1714
1714
3
BUILD SUCCESSFUL (total time: 0 seconds)
```

Quan s'utilitza la classe Object en Java per emmagatzemar elements de tipus diferents en una estructura de dades com un array, cal tenir en compte diversos comportaments i desafiaments.

En el cas de la inserció d'elements, es permet afegir elements de tipus diferents sense errors en temps de compilació, ja que tots els tipus són subtipus de la classe Object. Això significa que és possible incloure un String, un objecte Interval i un Boolean en la mateixa estructura sense problemes aparents.

No obstant això, en l'àmbit de mostrar informació amb un bucle, sorgeixen algunes dificultats. En intentar imprimir els elements amb un bucle, el mètode toString() de la classe Object es crida per defecte. La implementació predeterminada d'aquest mètode pot no proporcionar una representació significativa per a tots els tipus d'objectes, afectant potencialment la llegibilitat del codi.

Hi ha problemes potencials associats a la pèrdua de tipus segur. Es requereix fer casting explícit i gestionar possibles errors en temps d'execució quan s'intenta recuperar els elements amb els seus tipus reals. Aquesta pràctica pot fer que el codi sigui menys clar i més difícil de mantenir, ja que la informació de tipus específica es perd quan tots els elements s'emmagatzemen com a Object, podent conduir a un codi menys robust i més susceptible a errors en temps d'execució.

En resum, tot i que és possible emmagatzemar i accedir a elements de tipus diferents amb Object, aquest enfocament planteja desafiaments relacionats amb la pèrdua de tipus segur i la claredat del codi. En la pràctica, es recomana l'ús de generics o altres tècniques per mantenir la coherència de tipus quan es treballa amb estructures de dades en Java