IMPLEMENTACIÓN QUEUE

```java
package ut8problema1;
import java.util.ArrayList;
import java.util.Objects;

/**
 *
 * @author David García Valero
 */
public class Cua {

    // ATRIBUTOS

    private ArrayList<Object> array;

    // CONSTRUCTOR

    public Cua(ArrayList<Object> array) {
        this.array = array;
    }

    // MÉTODOS

    public boolean estabuida() {
        boolean buida = false;
        if (array.isEmpty()) {
            buida = true;
        }
        return buida;
    }

    public int getTamany() {
        int tamany = array.size();
        return tamany;
    }

    public Object cima() {
        Object objecte = array.getLast();
        return objecte;
    }

    public Object traure() {
        Object objecte = array.getFirst();
        array.removeFirst();
        return objecte;
    }

    public void afegir(Object objecte) {
        array.add(objecte);
    }
}

```

Lo primero que he hecho es buscar información detallada de cómo funciona un sistema queue en w3schools.

Una vez tuve claro que la los primeros objetos en ser añadidos tendrían que ser eliminados fui completando los métodos.

La mayoría de ellos utilizan métodos propios de ArrayList (isEmpty, size, getLast, add)

La parte que hace que funcione el queue es traure(), que obtiene el primer objeto, y después lo borra, tal y como dicta el esquema de las queue o colas.

```java
package ut8problema1;
import java.util.ArrayList;

/**
 *
 * @author David García Valero
 */
public class CuaTest {
    public static void main(String[] args) {
        ArrayList <Object> arrayCua = new ArrayList<>();
        Cua cua1 = new Cua(arrayCua);

        String cola1 = "Manuel";
        String cola2 = "Roberto";
        String cola3 = "Jose";
        String cola4 = "Lucía";

        cua1.afegir(cola1);
        cua1.afegir(cola2);
        cua1.afegir(cola3);
        cua1.afegir(cola4);

        Object cima1 = cua1.cima();
        System.out.println(cima1);

        System.out.println(cima1.toString().charAt(0));

        Object traer1 = cua1.traure();
        Object traer2 = cua1.traure();
        Object traer3 = cua1.traure();

        int tamany1 = cua1.getTamany();
        System.out.println(tamany1);

        System.out.println(cua1.estabuida());

        Object traer4 = cua1.traure();

        System.out.println(cua1.estabuida());

        Boolean colaBoolean1 = true;
        Boolean colaBoolean2 = false;

        Integer colaInteger1 = 10;

        cua1.afegir(colaBoolean1);
        cua1.afegir(colaBoolean2);
        cua1.afegir(colaInteger1);

        Object cima2 = cua1.cima();
        System.out.println(cima2);
    }
}
```

```
Output - UT8Problema1 (run)
run:
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Lucía
L
1
false
true
10
BUILD SUCCESSFUL (total time: 0 seconds)
```

En CuaTest inicializo un array queue y pruebo a añadir Strings, Booleans y Integer.

Me aseguro que los métodos creados funcionen, utilizándolos en distintos momentos específicos.

IMPLEMENTACIÓN STACK

```java
package ut8problema1;
import java.util.ArrayList;
import java.util.Objects;

/**
 *
 * @author David García Valero
 */
public class Pila {
    // ATRIBUTOS

    private ArrayList<Object> array;

    // CONSTRUCTOR

    public Pila(ArrayList<Object> array) {
        this.array = array;
    }

    // MÉTODOS

    public boolean estabuida() {
        boolean buida = false;
        if (array.isEmpty()) {
            buida = true;
        }
        return buida;
    }

    public int getTamany() {
        int tamany = array.size();
        return tamany;
    }

    public Object primer() {
        Object objecte = array.getFirst();
        return objecte;
    }

    public Object traure() {
        Object objecte = array.getLast();
        array.removeLast();
        return objecte;
    }

    public void afegir(Object objecte) {
        array.add(objecte);
    }
}

```

Busqué información de cómo era el stack en programación,
y los métodos que eran similares a la clase Cua, los copié tal y como estaban.

Los métodos que cambian son traure() y primer().
traure() recoge el último objeto y después lo borra y primer() trae el primer objeto en el array.

```java
package ut8problema1;
import java.util.ArrayList;

/**
 *
 * @author David García Valero
 */
public class PilaTest {
    public static void main(String[] args) {
        ArrayList <Object> arrayPila = new ArrayList<>();
        Pila pila1 = new Pila(arrayPila);

        String stack1 = "Manuel";
        String stack2 = "Roberto";
        String stack3 = "Jose";
        String stack4 = "Lucía";

        pila1.afegir(stack1);
        pila1.afegir(stack2);
        pila1.afegir(stack3);
        pila1.afegir(stack4);

        Object primer1 = pila1.primer();
        System.out.println(primer1);

        System.out.println(primer1.toString().charAt(0));

        Object traer1 = pila1.traure();
        Object traer2 = pila1.traure();
        Object traer3 = pila1.traure();

        int tamany1 = pila1.getTamany();
        System.out.println(tamany1);

        System.out.println(pila1.estabuida());

        Object traer4 = pila1.traure();

        System.out.println(pila1.estabuida());

        Boolean pilaBoolean1 = true;
        Boolean pilaBoolean2 = false;

        Integer pilaInteger1 = 10;

        pila1.afegir(pilaBoolean1);
        pila1.afegir(pilaBoolean2);
        pila1.afegir(pilaInteger1);

        Object primer2 = pila1.primer();
        System.out.println(primer2);
    }
}
```

```
Output - UT8Problema1 (run)
run:
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Manuel
M
1
false
true
true
BUILD SUCCESSFUL (total time: 0 seconds)
```

IMPLEMENTACIÓN OPCIONAL

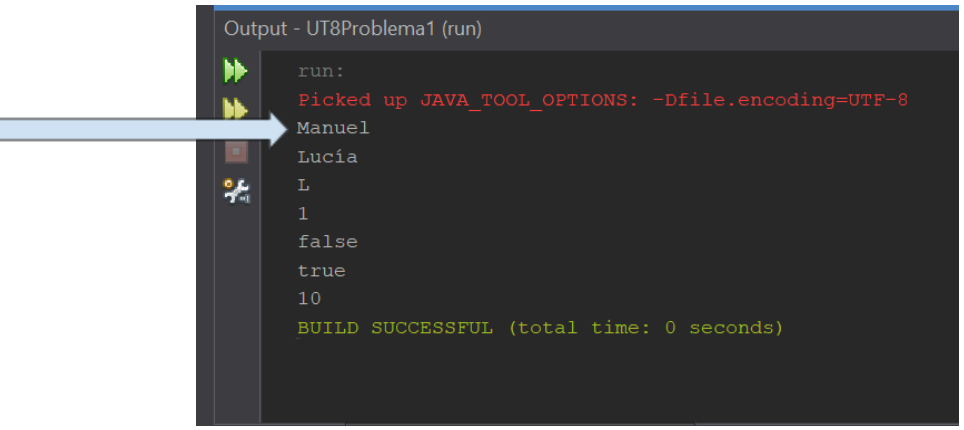
```
public Iterator traerIterador() {
    Iterator<Object> iterador = this.array.iterator();
    return iterador;
}
```

Creo un método que devuelve un iterador del array actual.

```
Iterator iterador = cua1.traerIterador();

System.out.println(iterador.next());
```

En una clase Test creo un iterador y pruebo su funcionalidad.



CONCLUSIÓN

Al utilizar ArrayList<Object> podemos generalizar nuestro array, para que así podamos decidir si queremos un array de Strings, Integers, Booleans, etc, e inicializarlo sin tener tantas capas de clases.

El problema es que si en algún momento existe confusión en un array y se añade un valor no deseado, puede llegar a dar problemas en entornos específicos.

Podríamos mejorar Pila y Cua añadiendo métodos de ordenación, alfabética en el caso de los String y de menor a mayor, en caso de los Integer.