Es proporciona un projecte de Netbeans, amb classes per a completar. La resolució dels exercicis és **individual** (si es fa en grup, procurar lliurar versions clarament diferenciades, doncs en altre cas es considera còpia i no puntua). S'ha de lliurar el projecte amb les classes completades abans del 29/09/2018.

L'objectiu d'aquest primer exercici és repassar els conceptes més bàsics de Java implementant una estructura de dades que serà útil més endavant a l'hora de programar protocols.

1.1 Cua circular indexada

Es vol implementar una cua circular indexada, que serà útil a l'hora de programar protocols. És una cua circular on la lectura (el get) és seqüència (com totes les cues que s'han vist fins ara), però l'escriptura (el put) és indexada, és a dir, es pot triar la posició on es posa l'element (permet suposar que els elements arriben desordenats). Per simplificar, se suposa que l'índex del primer element de la seqüència a guardar a la cua, és 0.

A l'hora de fer un put s'ha de saber si la posició està lliure (no és suficient comprovar només si la cua està plena, com fins ara). No és la millor solució, però simplifica molt la implementació, considerar que una posició de la cua està lliure sí el seu contingut és `null` (notar que les cues implementades a classe admeten `null` com a element).

En el següent codi es mostra la part de la implementació que té un comportament igual que el d'una cua circular (sense `put` indexat). El mètode `toString()` s'ha de completar.

```
package C19CuaIndexada;

public class CuaIndexada<T> implements Queue<T> {

    private final T[] buffer;
    private final int N;         //Grandària de la cua
    private int indexIni = 0;    //Índex relatiu inicial, per simplificar es considera 0
    private int posGet = 0;        //Posició actual de lectura
    private int numElems = 0; //Número d'elements a la cua

    public CuaIndexada(int N) {
        this.N = N;
        buffer = (T[]) (new Object[N]);
    }

    public void put(T t, int index) {
        //...
    }

    //put seqüèncial
    @Override
    public void put(T t) {
        if (isFull()) {
            throw new IllegalStateException("Cua plena");
        }
        buffer[(posGet + numElems) % N] = t;
        numElems = numElems + 1;
    }

    //put d'un array seqüèncial. Retorna el número d'elements que s'han posat.
    @Override
    public int put(T[] t, int offset, int len) {
```

```
    int i;
    for (i = 0; i < len; i++) {
        try {
            put(t[offset + i]);
        } catch (Exception e) {
            break;
        }
    }
    return i;
}

public int put(T[] t, int offset, int len, int index) {
    //...
}

@Override
public T get() {
    if (isEmpty()) {
        throw new IllegalStateException("Cua buida");
    }
    T t = null;
    t = buffer[posGet];
    buffer[posGet] = null;
    posGet = (posGet + 1) % N;
    numElems = numElems - 1;
    indexIni = indexIni + 1;
    return t;
}

@Override
public int get(T[] t, int offset, int len) {
    int i = 0;
    for (i = 0; i < len && !isEmpty(); i++) {
        t[offset + i] = get();
    }
    return i;
}

@Override
public boolean isFull() {
    return numElems == N;
}

//Indica si la posició actual del get conté algun element
@Override
public boolean isEmpty() {
    return buffer[posGet] == null;
}

@Override
public int size() {
    return numElems;
}

@Override
public String toString() {
    String str = "[";
    // Completar....
    // Es pot aprofitar que una posicio a null es considera buida
    //...
    return str;
}
```

```
}
```

Falta resoldre els mètodes de `put` indexats. El `put` indexat d'un únic element és el següent (s'ha de completar):

```java
public void put(T t, int index) {
    //Es genera excepció sí:
    //- l'index és menor que l'index inicial (representa elements ja rebuts)
    //- l'index és massa gran i per tant no correspon a cap posició de l'array.
    //...
    //Càlcul de la posició on s'ha de guardar l'element
    //...
    //Si la posició no està lliure es genera una excepció
    //...
    //Guardar l'element i actualitzar variables
    //...
}
```

Llavors, el `put` indexat d'ún array es pot construir a partir del `put` indexat d'un única element (completar).

```java
public int put(T[] t, int offset, int len, int index) {
    //Completar
    //...
}
```

Per comprovar que la implementació és correcta es pot executar la classe `Test.java`. Els resultat que s'ha d'obtenir per consola és del tipus (pot variar ja que és aleatori):

```
run:
cua::[0,1,2,3,4,5,6,7,8,9,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
rebuts::[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
cua::[_,_,_,_,_,30,31,32,33,34,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[_,_,_,_,_,30,31,32,33,34,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[_,_,_,_,_,30,31,32,33,34,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[_,_,_,_,_,30,31,32,33,34,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[_,_,_,_,_,30,31,32,33,34,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[_,_,_,_,_,30,31,32,33,34,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[_,_,_,_,_,30,31,32,33,34,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[_,_,_,_,_,30,31,32,33,34,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[_,_,_,_,_,30,31,32,33,34,10,11,12,13,14,15,16,17,18,19,_,_,_,_,_]
rebuts::[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
cua::[25,26,27,28,29,30,31,32,33,34,_,_,_,_,_,_,_,_,_,_,20,21,22,23,24]
rebuts::[20, 21, 22, 23, 24, 25, 26, 27, 28, 29]
rebuts::[30, 31, 32, 33, 34]
cua::[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,40,41,42,43,44,45,46,47,48,49]
cua::[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,40,41,42,43,44,45,46,47,48,49]
cua::[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,40,41,42,43,44,45,46,47,48,49]
cua::[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,40,41,42,43,44,45,46,47,48,49]
cua::[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,40,41,42,43,44,45,46,47,48,49]
cua::[_,_,_,_,_,_,_,_,_,_,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49]
cua::[_,_,_,_,_,_,_,_,_,_,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49]
rebuts::[35, 36, 37, 38, 39, 40, 41, 42, 43, 44]
cua::[50,51,52,53,54,55,56,57,58,59,_,_,_,_,_,_,_,_,_,_,45,46,47,48,49]
rebuts::[45, 46, 47, 48, 49, 50, 51, 52, 53, 54]
cua::[_,_,_,_,_,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,_,_,_,_,_]
cua::[75,76,77,78,79,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74]
rebuts::[55, 56, 57, 58, 59, 60, 61, 62, 63, 64]
cua::[75,76,77,78,79,_,_,_,_,_,_,_,_,_,_,65,66,67,68,69,70,71,72,73,74]
cua::[75,76,77,78,79,_,_,_,_,_,_,_,_,_,_,65,66,67,68,69,70,71,72,73,74]
rebuts::[65, 66, 67, 68, 69, 70, 71, 72, 73, 74]
cua::[75,76,77,78,79,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[75,76,77,78,79,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[75,76,77,78,79,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
rebuts::[75, 76, 77, 78, 79]
cua::[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
```

```
cua::[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]
cua::[_,_,_,_,_,_,80,81,82,83,84,85,86,87,88,89,_,_,_,_,_,_,_,_,_,_,_,_]
rebuts::[80, 81, 82, 83, 84, 85, 86, 87, 88, 89]
cua::[_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,90,91,92,93,94,95,96,97,98,99]
rebuts::[90, 91, 92, 93, 94, 95, 96, 97, 98, 99]
BUILD SUCCESSFUL (total time: 0 seconds)
```