Act 2.3.- Venta de entradas concierto

Clase EjercicioConcierto

```java
public class EjercicioConcierto {

    public static void main(String[] args) {

        Concierto concierto = new Concierto(entradas: 50);

        Random r = new Random();

        int nCompradores = r.nextInt(origin: 1, bound: 51);
        ArrayList<Comprador> arr = new ArrayList<Comprador>();

        for (byte i = 1; i < nCompradores; i++) {
            Comprador comprador = new Comprador(idComprador:i, color: "\u001B[32m", concierto);
            arr.add(e: comprador);
            comprador.start();
        }

        for(byte i = 0; i < arr.size(); i++){
            Comprador comprador = arr.get(index: i);
            try {
                comprador.join();
            } catch (InterruptedException ex) {
                System.out.printf(format: "ERROR: %s\n", args: ex.getMessage());
            }
        }

        System.out.printf(format: "Entradas restantes despues de las compras: %d\n", args: concierto.getEntradas());
    }

}
```

Clase Concierto

```java
public class Concierto {

    private int entradas;

    public Concierto(int entradas) {
        this.entradas = entradas;
    }

    public int getEntradas() {
        return entradas;
    }

    public synchronized boolean comprarEntrada(String color, int idCliente, int numerosEntradas) {

        if (entradasDisponibles(numerosEntradasCompradas: numerosEntradas)) {
            entradas -= numerosEntradas;
            System.out.printf(format: "%sCliente %d compra %d entradas. Entradas restantes: %d\n", args: color, args: idCliente, args: numerosEntradas, args: entradas);
            return true;
        } else {
            System.out.printf(format: "%sCliente %d intento comprar %d entradas, pero no hay suficientes disponibles.\n", args: color, args: idCliente, args: numerosEntradas);
            return false;
        }
    }

    public synchronized boolean entradasDisponibles(int numerosEntradasCompradas) {
        return this.entradas >= numerosEntradasCompradas;
    }

}
```

Clase Comprador

```java
public class Comprador extends Thread {

    private final int idComprador;
    private final String color;
    private final Concierto concierto;

    public Comprador(int idComprador, String color, Concierto concierto) {
        this.idComprador = idComprador;
        this.color = color;
        this.concierto = concierto;
    }

    @Override
    public void run() {
        try {
            Random r = new Random();
            concierto.comprarEntrada(color, idCliente: idComprador, numerosEntradas:r.nextInt(origin: 1,bound: 5));
            Thread.sleep((long) (r.nextInt(origin: 100, bound: 501)));
        } catch (InterruptedException ex) {
            System.out.printf(format: "ERROR: %s\n", args:ex.getMessage());
        }
    }

}
```

Resultado

```
Cliente 1 compra 4 entradas. Entradas restantes: 46
Cliente 20 compra 2 entradas. Entradas restantes: 44
Cliente 19 compra 2 entradas. Entradas restantes: 42
Cliente 18 compra 4 entradas. Entradas restantes: 38
Cliente 17 compra 3 entradas. Entradas restantes: 35
Cliente 16 compra 3 entradas. Entradas restantes: 32
Cliente 15 compra 2 entradas. Entradas restantes: 30
Cliente 14 compra 3 entradas. Entradas restantes: 27
Cliente 13 compra 1 entradas. Entradas restantes: 26
Cliente 12 compra 1 entradas. Entradas restantes: 25
Cliente 11 compra 4 entradas. Entradas restantes: 21
Cliente 10 compra 4 entradas. Entradas restantes: 17
Cliente 9 compra 2 entradas. Entradas restantes: 15
Cliente 8 compra 4 entradas. Entradas restantes: 11
Cliente 7 compra 1 entradas. Entradas restantes: 10
Cliente 6 compra 3 entradas. Entradas restantes: 7
Cliente 4 compra 4 entradas. Entradas restantes: 3
Cliente 5 compra 2 entradas. Entradas restantes: 1
Cliente 3 intento comprar 3 entradas, pero no hay suficientes disponibles.
Cliente 2 intento comprar 4 entradas, pero no hay suficientes disponibles.
Entradas restantes despues de las compras: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```