Listen

Am Nürnberger Flughafen starten und landen täglich viele Flugzeuge. Der Flughafen verfügt jedoch nur über eine einzige Start- bzw. Landebahn, sodass Starts und Landungen gemeinsam koordiniert werden müssen. Für die interne Verwaltung, welcher Flieger als nächstes bearbeitet werden soll, wird eine neue Software entwickelt. Sie haben die Aufgabe einen Teil dieser Software zu erstellen.

Das folgende UML-Klassendiagramm gibt einen Überblick über den für Sie relevanten Teil der Software:

Implementieren Sie die beiden Methoden addNewFlight(Flug) und addEmergency(Flug) der Klasse Warteschlange!

- In der Methode `addNewFlight`
  - soll ein neu übergebener Flug der Warteschlange hinzugefügt werden.
- In der Methode `addEmergency`
  - soll für den übergebenen Flug überprüft werden, ob der Flug bereits in der Warteschlange vorkommt oder ob es sich um einen komplet neuen Flug handelt.
  - soll der Notfall die erste Priorität in der Warteschlange erhalten, d.h. zwingend als nächstes abgearbeitet werden.
  - soll die korrekte Funktionalität der Warteschlange weiterhin gegeben sein.

Jeder Flug wird in der Warteschlange mit einem eigenen Ticket verwaltet. Die Funktionalitäten der Tickets und der Flüge entnehmen Sie dem Quelltext.

Klasse „Flug“

```
3  public class Flug {
4    private int flugnummer;
5    private String startFlughafen;
6    private String zielFlughafen;
7
8    public Flug(int flugnummer, String startFlughafen, String zielFlughafen) {
9      this.flugnummer = flugnummer;
10     this.startFlughafen = startFlughafen;
11     this.zielFlughafen = zielFlughafen;
12   }
13
14   public int getFlugnummer() {
15     return flugnummer;
16   }
17
18   public String getStartFlughafen() {
19     return startFlughafen;
20   }
21
22   public String getZielFlughafen() {
23     return zielFlughafen;
```

```
  }
}
```

Klasse „Ticket“

```
public class Ticket {
  public Flug flug;
  public Ticket next;
  public boolean startetInNBG = false;
  public boolean landetInNBG = false;

  public Ticket(Flug flug) {
    this.flug = flug;
    if (flug.getStartFlughafen().equals("NUE")) {
      startetInNBG = true;
    } else {
      landetInNBG = true;
    }
  }
}
```

Klasse „Warteschlange“

```
public class Warteschlange {
  Ticket first = null;

  public void addNewFlight(Flug flug) {
    Ticket counter = first;
    if (first == null) {
      first = new Ticket(flug);
    } else {
      while (counter.next != null) {
        counter = counter.next;
      }
      counter.next = new Ticket(flug);
    }
  }

  public void addEmergency(Flug flug) {
    if (first.flug.getFlugnummer() == flug.getFlugnummer()) {
      return;
    }
    Ticket counter = first;
    Ticket counter2 = first.next;
    Ticket emergency = null;
    while (counter2 != null) {
      if (counter2.flug.getFlugnummer() == flug.getFlugnummer()) {
        emergency = counter2;
        counter.next = counter2.next;
        break;
      } else {
        counter = counter2;
        counter2 = counter2.next;
      }
    }
    if (emergency == null) {
      emergency = new Ticket(flug);
    }
```

```
38        emergency.next = first;
39        first = emergency;
40      }
41    }
```