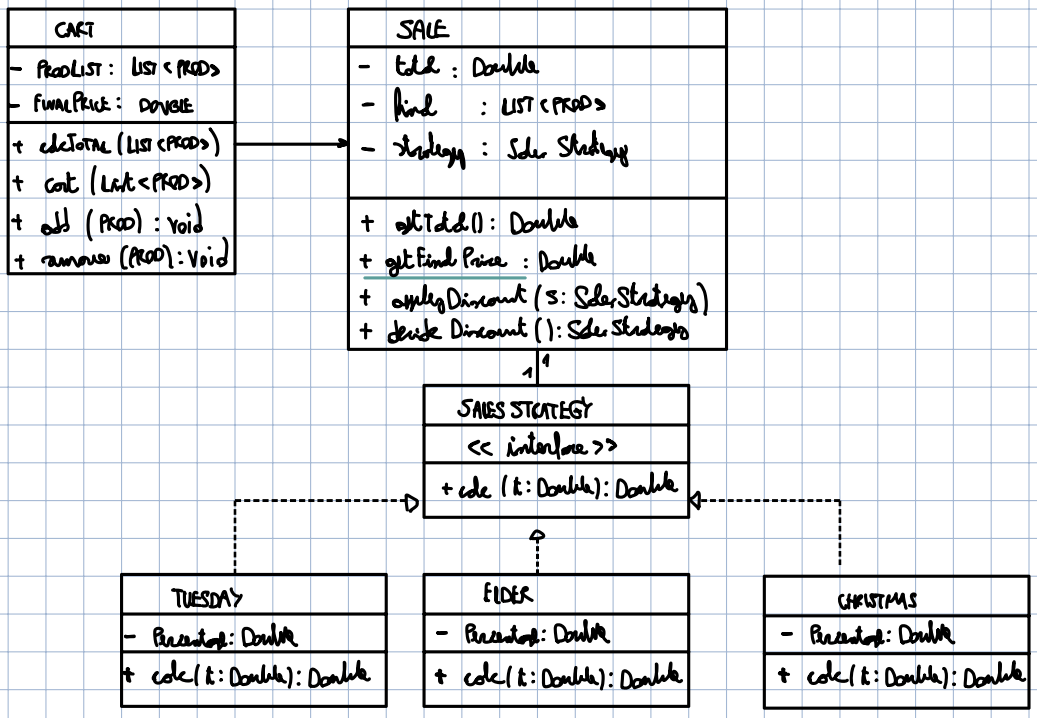
2021-07-15

E1

```
public class MusicLibrary {
  // Ritorna l'insieme di artisti che eseguono almeno un brano nella collezione.
  public /*@ pure @*/ Set<Artist> getArtists();

  // Ritorna la lista di brani eseguiti dall'artista.
  // Lancia una UnknownArtistException se la collezione non contiene brani di artist.
  public /*@ pure @*/ List<Song> getByArtist(Artist artist) throws UnknownArtistException;

  // Ritorna l'insieme dei brani eseguiti in una certa data.
  // Lancia una UnknownDateException se nessun brano e' stato eseguito in quella data.
  public /*@ pure @*/ Set<Song> getByDate(Date date) throws UnknownDateException;

  // Aggiunge il brano alla collezione.
  public void addSong(Song s);
}

public /*@ pure @*/ class Song {
  // Ritorna l'artista che esegue il brano
  public Artist getArtist();

  // Ritorna la data in cui e' stato eseguito il brano
  public Date getDate();
```

```
/*@ requires date != null
  @
  @ ensures (\forall int i; i>=0 && i < this.getArtists().size;
  @          (\forall int j; j>=0 && j < this.getByArtist(this.getArtists().get(i));
  @           !(\exists int k; k>=0 && k<\result.size();
  @              this.getByArtist(this.getArtists().get(i)).get(j).getDate.equals(date) &&
  @               !(\result.contains(this.getByArtist(this.getArtists().get(i)).get(j))
  @
  @ signals (UnknownDateException UDE) (\forall int i; i>=0 && i< this.getArtists().size();
  @                                     !(\exists int j; j>=0 && j< this.getByArtist(this.getArtists().get(i)).size()
  @                                        this.getByArtist(this.getArtists().get(i)).get(j).getDate().equals(date)
  @
  @ signals (NullPointerException NPE) date == null
*/


/*@ @ private invariant set != null &&
  @ !set.contains(null) &&
  @ (\forall int i; i>=0 && i<set.size();
  @  !(\exists int j; j>i && j<set.size();
  @      set.get(i).equals(set.get(j));
  @  )
  @ )
*/
```

E2



E3

- ```
  public static List<Person> eldest (List<Person> people) {
      return people.stream().sorted((p1,p2) -> p1.getAge().compareTo(p2.getAge())).findFirst()
  }
  ```

  ```
  { return people.stream().filter(p -> people.stream().noMatch(p2 -> p2.getAge() > p.getAge()))
                  .collect(Collectors.toList())
  }
  ```

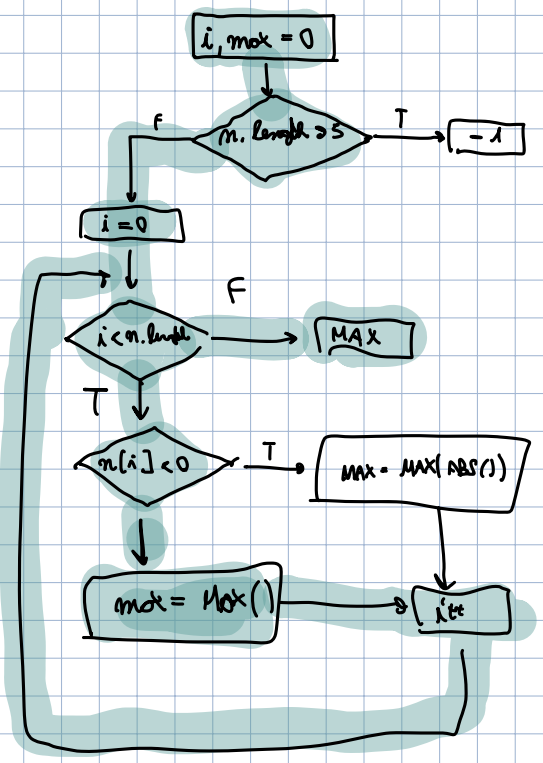
  ```
  public static List<Person> elder (List<Person> people) {
      List<Person> elder = people.stream().filter(p -> p.getAge() > 60)
                                  .collect(Collectors.toList())
  }
  ```

- ```
  public static void longest (List<String> names) {
      final Optional<String> theLongest = names.stream().map(c -> c.chars().count())
                                               .filter(n -> names.stream()
                                                   .noMatch(n2 -> n2.chars().count() > n))
                                               .collect(Collectors.toList())
      System.out.println(theLongest.isPresent() ? theLongest : "null")
  ```

E4

- for(;;) non dà errore in compilazione, ma è equivalente a while(true)
- Int non è un oggetto che implementa Runnable, probabilmente si intendeva Thread
- Potrebbe
- No → sì è volatile
- Sì

E5



T1 : {0 0 0 0 0} → -1 (9/10 ; 8/11)

T2 : {1 2 3 4 5} → 5 (9/10 ; 8/11)

T3 : {-1 -2 -3 -4 -5} → 5 (8/10 ; 8/11)

T4 : {1 2 3 4 5 6} → -1 (3/10 ; 2/11)

T5 : {-10 10 3 5 -6} → 10 (9/10 ; 10/11)

T6 : {} → 0