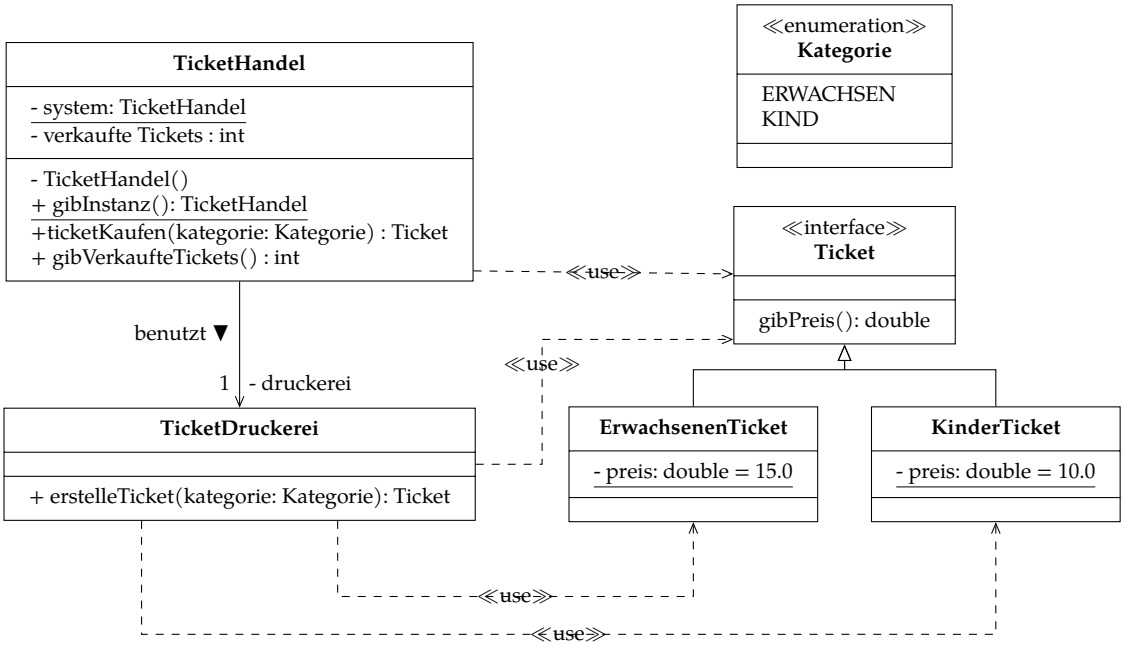
Aufgabe 3



Ihnen sei ein UML-Klassendiagramm zu folgendem Szenario gegeben. Ein Benutzer (nicht im Diagramm enthalten) kann über einen `TicketHandel` Tickets erwerben. Dabei muss der Benutzer eine der zwei Ticketkategorien angeben. Das Handelsystem benutzt eine `TicketDruckerei`, um ein passendes `Ticket` für den Benutzer zu erzeugen.

(a) Im angegebenen Klassendiagramm wurden zwei unterschiedliche Entwurfsmuster verwendet. Um welche Muster handelt es sich? Geben Sie jeweils den Namen des Musters sowie die Elemente des Klassendiagramms an, mit denen diese Muster im Zusammenhang stehen. ACHTUNG: Es handelt sich dabei *nicht* um das *Interface-* oder das *Vererbungs*muster.

Einzelstück (Singleton): `TicketHandel`

Einzelstück (Singleton) stellt eine statische Methode bereit, mit deren Hilfe die Klienten nur auf eine einzige Instanz der Klasse zugreifen können.

(b) Nennen Sie zwei generelle Vorteile von Entwurfsmustern.

- Wiederverwendung einer bewährten Lösung für eine bestimmte Problemstellungen
- Verbesserung der Kommunikation unter EntwicklerInnen

(c) Geben Sie eine Implementierung der Klasse `TicketHandel` an. Bei der Methode `ticketKaufen()` wird die Anzahl der verkauften Tickets um 1 erhöht und ein entsprechendes Ticket erstellt und zurückgegeben. Beachten Sie den Hinweis auf der nächsten Seite.

```
3  public class TicketHandel {
4    private static TicketHandel system;
5    private int verkaufteTickets = 0;
6    private TicketDruckerei druckerei;
7
8    private TicketHandel() {
9      druckerei = new TicketDruckerei();
10   }
11
12   public static TicketHandel gibInstanz() {
13     if (system == null) {
14       system = new TicketHandel();
15     }
16     return system;
17   }
18
19   public Ticket ticketKaufen(Kategorie kategorie) {
20     verkaufteTickets++;
21     return druckerei.erstelleTicket(kategorie);
22   }
23
24   public int gibVerkaufteTickets() {
25     return verkaufteTickets;
26   }
27 }
```

github: raw

(d) Geben Sie eine Implementierung der Klasse `TicketDruckerei` an.

```
3  public class TicketDruckerei {
4    public Ticket erstelleTicket(Kategorie kategorie) {
5      if (kategorie == Kategorie.ERWACHSEN) {
6        return new ErwachsenenTicket();
7      } else {
8        return new KinderTicket();
9      }
10   }
11 }
```

github: raw

(e) Geben Sie eine Implementierung der Klasse `KinderTicket` an.

```
3  public class KinderTicket implements Ticket {
4    private static double preis = 10.0;
5
6    public double gibPreis() {
```

```
7       return preis;
8     }
9   }
```

github: raw

Hinweis: Die Implementierungen *müssen* sowohl dem Klassendiagramm, als auch den Konzepten der verwendeten Muster entsprechen. Verwenden Sie eine objektorientierte Programmiersprache, vorzugsweie *Java*. Sie müssen sich an der nachfolgenden Testmethode und ihrer Ausgabe orientieren. Die Testmethode muss mit Ihrer Implementierung ausführbar sein und sich semantisch korrekt verhalten.

Quelltext der Testmethode:

```
4   public static void main(String[] args) {
5     TicketHandel.gibInstanz().ticketKaufen(Kategorie.ERWACHSEN);
6     TicketHandel.gibInstanz().ticketKaufen(Kategorie.KIND);
7     System.out.println("Anzahl verkaufter Tickets: " +
      ↪  TicketHandel.gibInstanz().gibVerkaufteTickets());
8   }
```

github: raw

Konsolenausgabe:

Anzahl verkaufter Tickets: 2