BORD-Festival-Management-System Abschlusspräsentation

Olga Klassen, Benjamin Swarovsky, Raphael Freybe, Daniel Depta

Fachhochschule Erfurt, 03.02.2020

SOLL VS IST

Ursprünglich geplant

- Tickets verkaufen
- Kundenaccount anlegen
- Eventverwaltung
- Bühnen- und Bandverwaltung

Umgesetzt

- Tickets verkaufen ✓
- Kundenaccount anlegen ✓
- Eventverwaltung ✓
- Bühnen- und Bandverwaltung ✓

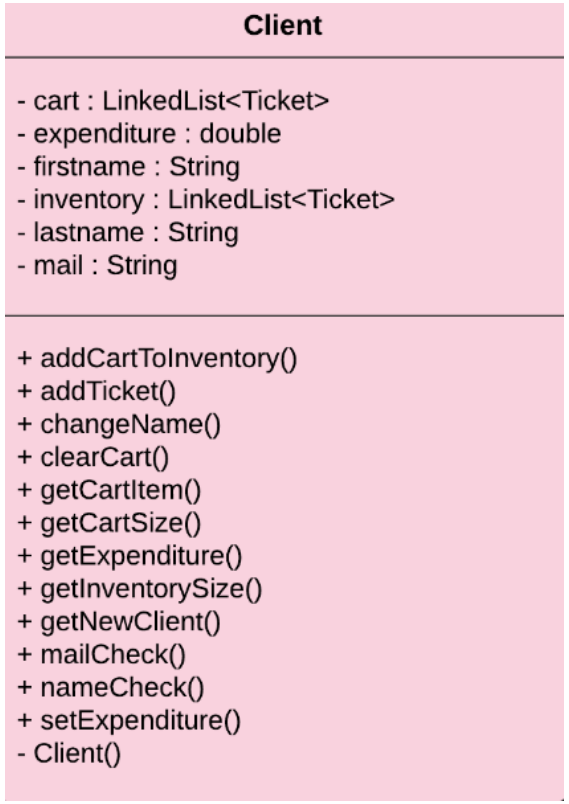
zusätzlich realisiert

- Festivaleinnahmen, Festivalausgaben
- Kartenkontingent

FEATURES - CLIENT

Client

- Ändern persönlicher Daten
- Validierung
- Warenkorb, Inventar und Ausgaben
- Hinzufügen von Tickets




```
public void addTicket(Ticket.TicketType type, TicketManager ticketmanager)
        throws TicketNotAvailableException
{
    if(!ticketmanager.isAvailable(type, numberOfCartTickets: 1)) {
        throw new TicketNotAvailableException("No more tickets available");
    }
        if(ticketmanager.getTicket(type)!= null) {
            this.cart.add(ticketmanager.getTicket(type));
        }
}
```

FEATURES – BÜHNE UND BAND

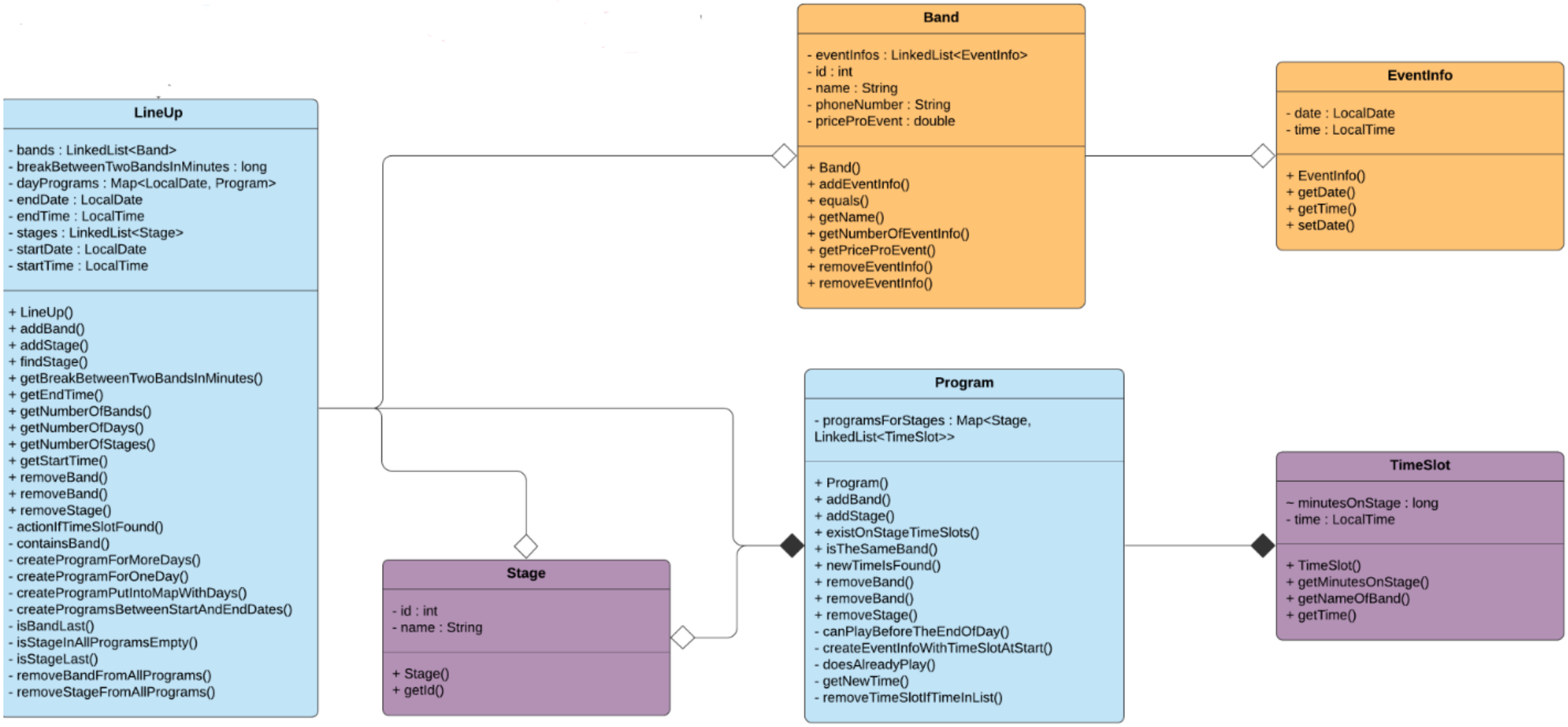
- Anpassung des Eventbudgets beim hinzufügen von Bands
- Hinzufügen und löschen von:

  -Bühnen, Timeslots und Bands

```
private void doesAlreadyPlay(Band band, LocalTime time) throws TimeSlotCantBeFoundException {

    for (Map.Entry<Stage, LinkedList<TimeSlot>> entry : programsForStages.entrySet()) {
        for (int i = 0; i < entry.getValue().size(); i++) {
            LocalTime timeInTimeSlot = entry.getValue().get(i).getTime();
            String nameOfBandInTimeSlot = entry.getValue().get(i).getNameOfBand();
            //if the same time and band name exist on another stage
            if ((time.compareTo(timeInTimeSlot) == 0) && (band.getName().equals(nameOfBandInTimeSlot))) {
                throw new TimeSlotCantBeFoundException("This band plays already on another stage");
            }
        }
    }
}
```

FEATURES – BÜHNE UND BAND



FEATURES - TICKET

- Verwaltung von Preisstufen
- Auswertung des Ticketkontingents
- Anpassung der Festivaleinnahmen nach Ticketverkauf
- Anpassung des Ticketkontingents nach Ticketverkauf

```
    } else if(ticketType == Ticket.TicketType.VIP) {
        if (isAvailable(ticketType, numberOfVipTicketsSold)) {
            numberOfVipTicketsSold++;
            ticketIncome += client.getCartItem(index).getStdPrice();
        } else {
            throw new TicketNotAvailableException("Not enough VIP-tickets available");
        }
    }
    index++;
}
this.numberOfDayTicketsLeft -= numberOfDayTicketsSold;
this.numberOfCampingTicketsLeft -= numberOfCampingTicketsSold;
this.numberOfVipTicketsLeft -= numberOfVipTicketsSold;

client.addCartToInventory();
client.clearCart();
updateIncomeTicketSales(ticketIncome);
client.setExpenditure(ticketIncome);

if(automaticPriceLevelChange){
    updatePriceLevel();
}
```

FEATURES - TICKET

PriceLevel

- CampingTicketPrice : double
- PercentageForPriceLevel : double
- VipTicketPrice : double
- dayTicketPrice : double

+ PriceLevel()
+ compareTo()
+ getCampingTicketPrice()
+ getDayTicketPrice()
+ getPercentageForPriceLevel()
+ getVipTicketPrice()

TicketManager

- actualPriceLevel : int
- automaticPriceLevelChange : boolean
- incomeTicketSales : double
- numberOfCampingTickets : int
- numberOfCampingTicketsLeft : int
- numberOfDayTickets : int
- numberOfDayTicketsLeft : int
- numberOfVipTickets : int
- numberOfVipTicketsLeft : int
- priceLevels : ArrayList<PriceLevel>

+ TicketManager()
+ getActualPriceLevelIndex()
+ getAutomaticPriceLevelChange()
+ getIncomeTicketSales()
+ getNumberOfCampingTickets()
+ getNumberOfCampingTicketsLeft()
+ getNumberOfDayTickets()
+ getNumberOfDayTicketsLeft()
+ getNumberOfSoldCampingTickets()
+ getNumberOfSoldDayTickets()
+ getNumberOfSoldVipTickets()
+ getNumberOfVipTickets()
+ getNumberOfVipTicketsLeft()
+ getPercentageForPriceLevel()
+ getTicket()
+ isAvailable()
+ sellTickets()
+ setAutomaticPriceLevelChange()
+ setPriceLevel()
+ setTicketDescription()
+ setTicketStdPrice()
+ totalNumberOfSoldTickets()
+ totalNumberOfSoldTicketsInPercent()
+ totalNumberOfTickets()
+ totalNumberOfTicketsLeft()
- PriceLevelIndexInOnlyValueArea()
- isPercentageOfSoldTicketsExceededAndIsTheNextPriceLevelExisting()
- isPriceLevelUpdateManualAndThePriceLevelIndexInOnlyValueArea()
- setTicketPrices()
- updateIncomeTicketSales()
- updatePriceLevel()

NÜTZLICHE TOOLS

- Tests (Beispiele)

```
@Test
void should_throw_exception_for_mail_with_missing_at_symbol() throws MailException {
    String testmail = "testtest.de";
    assertThrows(MailException.class, () -> {
        Client client = Client.getNewClient(firstname, lastname, testmail, id, address);
    });
}

@Test
void should_throw_exception_for_mail_with_missing_domain() throws MailException {
    String testmail = "test@test";
    assertThrows(MailException.class, () -> {
        Client client = Client.getNewClient(firstname, lastname, testmail, id, address);
    });
}
```

- Benutzte Regular Expression:

```
"^[\\w!#$%&'*+/=?_`(){|}~\"@<>,:;^-]+(?:\\.[\\w!#$%&'*+/=?_`(){|}~\"@<>,:;^-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}$"
```

NÜTZLICHE TOOLS

- Parametrisierte Tests

```
@ParameterizedTest
@ValueSource(strings = { "John..Doe@example.com",
                         "john.doe@example..com",
                         " johndoe@example.com",
                         "johndoe@example.com ",
                         "valid.mail@example.com",
                         "a\"b(c)d,e:f;gi[j\\k]l@example.com" })
```

- ✔ [1] John..Doe@example.com
- ✔ [2] john.doe@example..com
- ✔ [3] johndoe@example.com
- ✔ [4] johndoe@example.com
- ✘ [5] valid.mail@example.com
- ✔ [6] a"b(c)d,e:f;gi[j\k]l@example.com

```
void should_throw_exception_for_invalid_mail(String input) throws MailException {
    assertThrows(MailException.class, () -> {
        Client client = Client.getNewClient(firstname, lastname, input, id, address);
    });
}
```

NÜTZLICHE TOOLS

- Interfaces

```
public interface ITicketManager {

    public Ticket getTicket(Ticket.TicketType type);

    public boolean isAvailable(Ticket.TicketType type, int numberOfCartTickets);

    public void setTicketDescription(String description, Ticket.TicketType type);

    public void setTicketStdPrice(double stdPrice, Ticket.TicketType type);

    public void setAutomaticPriceLevelChange(boolean isPriceLevelChangeAutomatic);

    // ...
```

NÜTZLICHE TOOLS

- Factorys

```
public static Event getNewEvent(LocalDate startDate, LocalDate endDate, String name, double budget,
                                 int maxCapacity, Stage stage, TicketManager ticketManager, Address address)
                                 throws DateDisorderException {
    if (endDate.isBefore(startDate)) {
        throw new DateDisorderException("End date can't be before start date");
    }

    return new Event(startDate, endDate, name, budget, maxCapacity, stage, ticketManager, address);

}
```

LESSONS LEARNED

- Mindestens 1x wöchentlich Treffen
- (Parametrisierte) Tests
- Umgang mit git/Github
  - Regelmäßig pushen und pullen
  - Issues für Aufgabenteilung
- Maven und Github Actions sind hilfreich

155 erfolgreiche Testfälle

✓ **bord-festival**
on: push (b66992a)

✓ **bord-festival**
on: push (fe5ac8e)

✗ **bord-festival**
on: push (8e055dc)

✓ **bord-festival**
on: push (4063bcf)

✓ **bord-festival**
on: push (082c657)

bord-festival passing

DEMNÄCHST IM KINO

Front-End:

- Benutzerrollen: Admin, Kunde
- Ticketshop
- Admin- und Kunden-Accountverwaltung
- Veranstaltungsübersicht, Veranstaltungsliste verwalten
- Anzeige des Line-Ups

Vielen Dank für Ihre Aufmerksamkeit!

https://github.com/fh-erfurt/bord-festival