**Exercise A2**

**1. Software Use-Case Descriptions**

**1.1. Use Case UC1: Sell Ticket**

**Use Case ID:** UC1  
**Name:** Sell Ticket  
**Primary Actor:** Cashier  
**Stakeholders & Interests:**

* **Visitor:** wants a correct ticket with applicable discounts and any special-attraction add-ons.
* **Zoo Management:** wants accurate revenue recording and enforcement of discount rules.

**Preconditions:**

* Cashier is authenticated and the system is online.

**Postconditions:**

* One or more Ticket records (including any reptilarium/aquarium add-on tickets) are stored.
* A Payment record is stored and linked to the ticket(s).
* Tickets are printed for the visitor.

**Trigger:**

* Cashier selects “Sell Ticket” in the application.

**Main Success Scenario:**

1. Cashier enters visitor category (Adult, Student, Soldier, Retiree or Group) and, if Group, the group size.
2. System computes base price and applies any discount.
3. Cashier optionally adds reptilarium and/or aquarium tickets.
4. Cashier selects payment method (Cash or Credit) and enters required payment details.
5. System processes the payment and confirms success.
6. System creates the Ticket(s), the Payment, prints the ticket(s), and displays a confirmation.

**Extensions:**

* **2a. Invalid category**: System displays error message “Invalid visitor category” and returns to step 1.
* **4a. Credit card declined**: System displays “Payment declined,” and cashier may retry or switch to cash.
* **5a. Partial payment for group**: System prompts for remaining amount; cashier collects and returns to step 5.

**1.2. Use Case UC2: Relocate Animal**

**Use Case ID:** UC2  
**Name:** Relocate Animal  
**Primary Actor:** Administrator  
**Stakeholders & Interests:**

* **Zoo Management:** needs accurate records of animal locations.
* **Animal Care Staff:** requires assurance that paddock capacity and compatibility rules are enforced.

**Preconditions:**

* Administrator is authenticated.
* The animal exists and has a current paddock assignment.
* Target paddock exists and has available capacity.

**Postconditions:**

* Animal’s defaultLocation is updated to the new paddock.
* A MovementHistory record is created with timestamp, source paddock, and destination paddock.

**Trigger:**

* Administrator selects an animal record and chooses “Relocate” in the UI.

**Main Success Scenario:**

1. System displays the animal’s current paddock and a list of available target paddocks.
2. Administrator selects the destination paddock.
3. System validates capacity and, if required, zone compatibility.
4. Administrator confirms the relocation.
5. System updates the Animal record’s defaultLocation and creates a MovementHistory entry.
6. System displays a relocation success message.

**Extensions:**

* **3a. Insufficient capacity**: System displays “Selected paddock is full,” and returns to step 2.
* **3b. Zone incompatibility**: System warns “Zone mismatch—confirm override?”; if overridden, proceed; otherwise, return to step 2.

**2. System Operations**

| **Operation Name** | **Signature** |
| --- | --- |
| **sellTicket** | sellTicket(visitorType: VisitorType, groupSize: Integer, addReptilarium: Boolean, addAquarium: Boolean, paymentMethod: PaymentMethod, paymentDetails: PaymentDetails): TicketReceipt |
| **relocateAnimal** | relocateAnimal(animalID: String, targetPaddockID: String): RelocationReceipt |

**3. Operation Contracts**

**3.1. sellTicket**

Operation:

sellTicket(visitorType, groupSize, addReptilarium, addAquarium, paymentMethod, paymentDetails)

: TicketReceipt

Preconditions:

• visitorType ∈ {ADULT, STUDENT, SOLDIER, RETIREE, GROUP}

• if visitorType = GROUP, then groupSize ≥ 2

• paymentMethod ∈ {CASH, CREDIT}

• paymentDetails ≠ null and valid for the chosen paymentMethod

Postconditions:

Let basePrice = ZOO.BASE\_PRICE × groupSize

Let discountFactor = getDiscount(visitorType, groupSize)

Let addonPrice = (addReptilarium ? ZOO.REPTILARIUM\_PRICE : 0)

+ (addAquarium ? ZOO.AQUARIUM\_PRICE : 0)

totalPrice = (basePrice × (1 – discountFactor)) + addonPrice

• A new `Ticket` T is created with T.price = totalPrice and stored.

• A new `Payment` P is created with P.amount = totalPrice,

P.method = paymentMethod, and linked to T.

• System prints the ticket(s).

• Returns a `TicketReceipt` summarizing T and P.

Exceptions:

• `InvalidVisitorTypeException` if visitorType not recognized

• `InvalidGroupSizeException` if GROUP and groupSize < 2

• `PaymentFailedException` if paymentMethod = CREDIT and authorization fails

**3.2. relocateAnimal**

Operation:

relocateAnimal(animalID, targetPaddockID)

: RelocationReceipt

Preconditions:

• animalID corresponds to an existing `Animal` A

• targetPaddockID corresponds to an existing `Paddock` P

• P.capacityRemaining ≥ 1

• (Optional) A.species.zoneOfOrigin = P.zoneOfOrigin

or an override flag is set by the administrator

Postconditions:

• Old paddock’s capacityRemaining is incremented by 1

• P.capacityRemaining is decremented by 1

• A.defaultLocation ← targetPaddockID

• A new `MovementHistory` M is created with:

– M.animalID = animalID

– M.from = oldPaddock

– M.to = targetPaddockID

– M.timestamp = current time

• All changes are persisted

• Returns a `RelocationReceipt` summarizing the move

Exceptions:

• `AnimalNotFoundException` if no Animal with animalID

• `PaddockNotFoundException` if no Paddock with targetPaddockID

• `InsufficientCapacityException` if P.capacityRemaining < 1

• `ZoneMismatchException` if zones differ and no override flag