MONASH University

Pass Task 3.1 DownTown Bikes & Bike Rental System Case Study

Summary

Complete design specification for a case study (see appendix I).

Purpose: To become familiar with sequence and class diagram.

Task: Develop a **complete design specification** (using UML and OO methods) for the system. You must work in a group of 3-4 students. The complete design specification will include the application design (sequence and class diagrams are the major components), database design, interface design, architecture design and network design.

Time: This task should be started during your week 3 studio and completed during the week, by Friday, ready for feedback

Resources: Week 3 lecture material.
Week 3 studio presentation
Tutor in Studio Session
Study partners in class
Help desk (from week 3)

Feedback: Once submitted to Doubtfire, the tutor will give you some feedback on the complete design specification**.**

Next: Move onto Task 4.1.

Pass Task 3.1 - Submission details and assessment criteria

Make sure you include the correct Task3.1.pdf file of your **complete design specification**. If you have used the word file, then create a pdf of your solution.

Appendix I:

DownTown Bikes & Bike Rental System Case Study

A system is being developed to support core business operations of the DownTown chain of Bike stores. The analysis phase has ended and the design phase is about to begin. You have been contracted to complete the design specification and implementation plan. The following background information will help you to understand the results of the analysis phase and the scope and nature of the system you have to design.

DownTown Bikes is a chain of 11 bike stores scattered throughout a major metropolitan area in the Melbourne. The chain started with a single store several years ago and has grown to its present size. Paul and Pat Lowes, the owners of the chain, know that to compete with the national chains requires a state-of-the-art stock management and customer relationship system. You have been asked to develop the system requirements.

Each store has a stock of bikes for sale and bikes for day rent. It is important to keep track of each bike in stock to know and to identify its category (electric, racing, mountain, road, fold-up and so on); its user type (child, mens, womens etc), size (cm), and other general information such as; brand, year of manufacture, cost, and so forth. In addition to tracking each bike for sale, the business must track each individual stock item to note its purchase date, its condition, and its rental status. User functions must be provided to maintain this inventory information.

Customers, the lifeblood of the business, are also tracked. DownTown considers individuals as well as each family to be a customer, so special mailings and promotions are offered to each household. For any given customer, any person in the same household may be eligible for discounts and special offers. All bike purchasers receive a loyalty card which provides for free servicing for the first three months and 20% off the cost of general servicing for the next 12 months. However, these offers are only available to customers over 18 years of age.

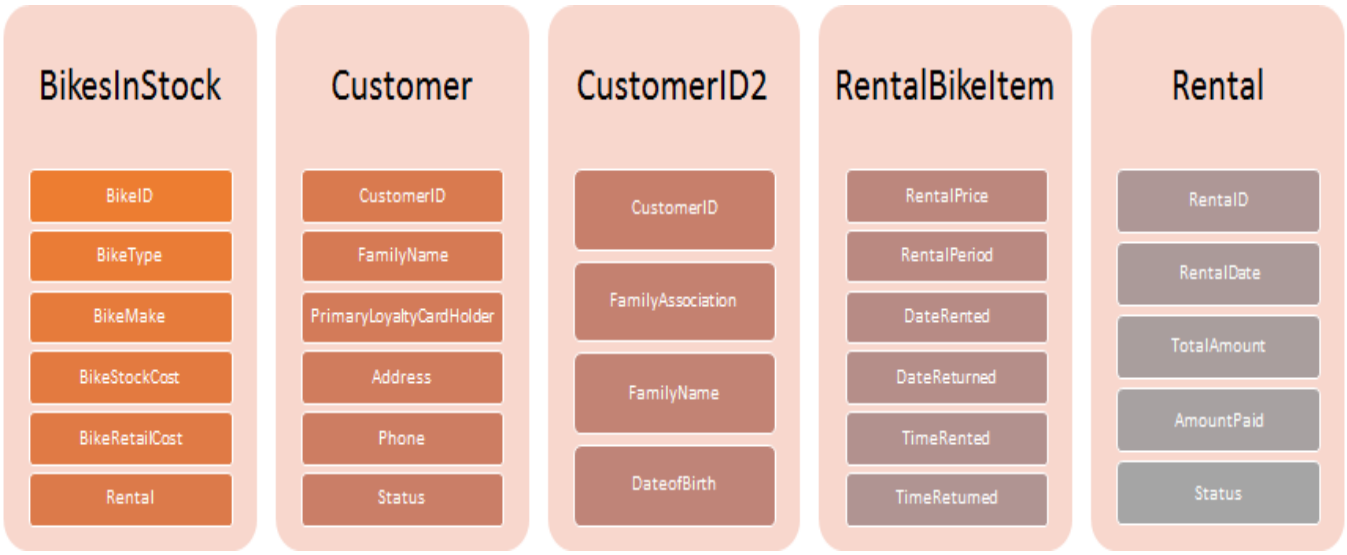
Each time a bike is rented, the system must keep track of which bike is rented; the rental date and time and the return date and time; and the household and person renting the bike. As tourists often rent bikes for occasional use the system must be able to cope with this.

Bikes can also be provided as a rental bike for a one week trial period prior to purchase for loyalty customers. Customers pay for rentals when checking out the bikes and leave a deposit for insurance. Paul and Pat also offer a high end service for elite cyclists in some stores and are looking to expand this offering.

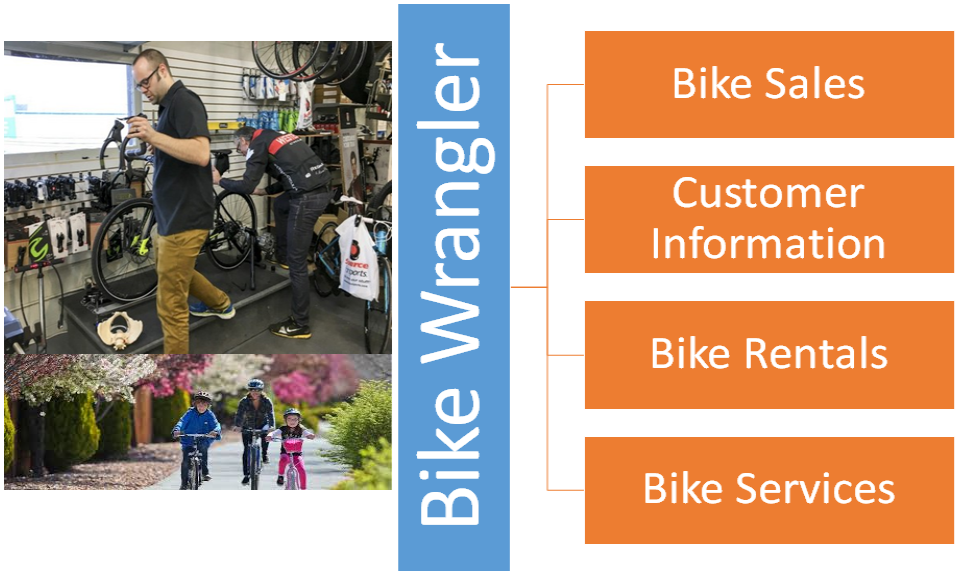
Paul and Pat are wanting to adopt a more efficient cloud based service across their stores, to build customer loyalty through social media and to develop a smart phone app for booking rentals, monitoring their bike locations and to encourage bike renters and purchasers to engage through their facebook and instagram site.

The following figure shows the first stage **domain class diagram** created by the analysts who abandoned the project after their startup in pressed juices took off during the analysis phase of the system development.

Redraw this example analysis showing the relationships between the **domain classes** identified by the developers.
Consider any additions needed and remove any you see as unnecessary



The following figure shows the **use case** diagram created by the analysts during the analysis phase of the system development.

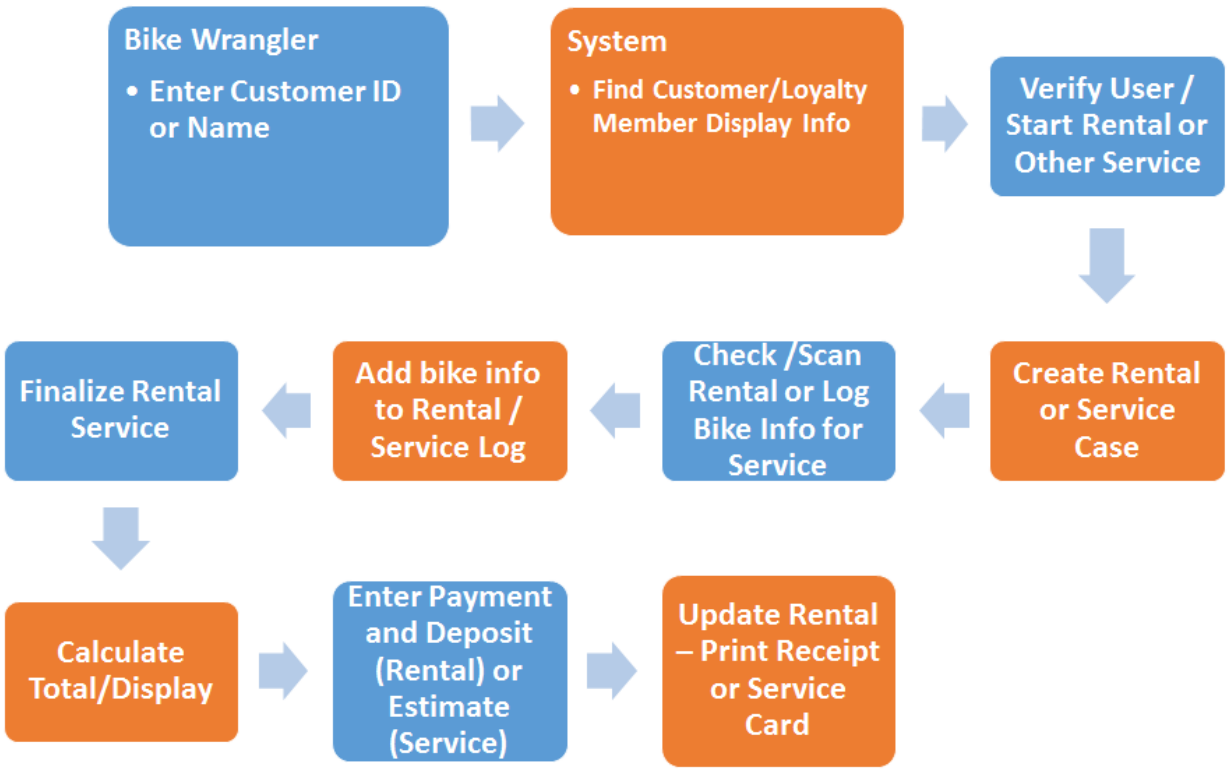


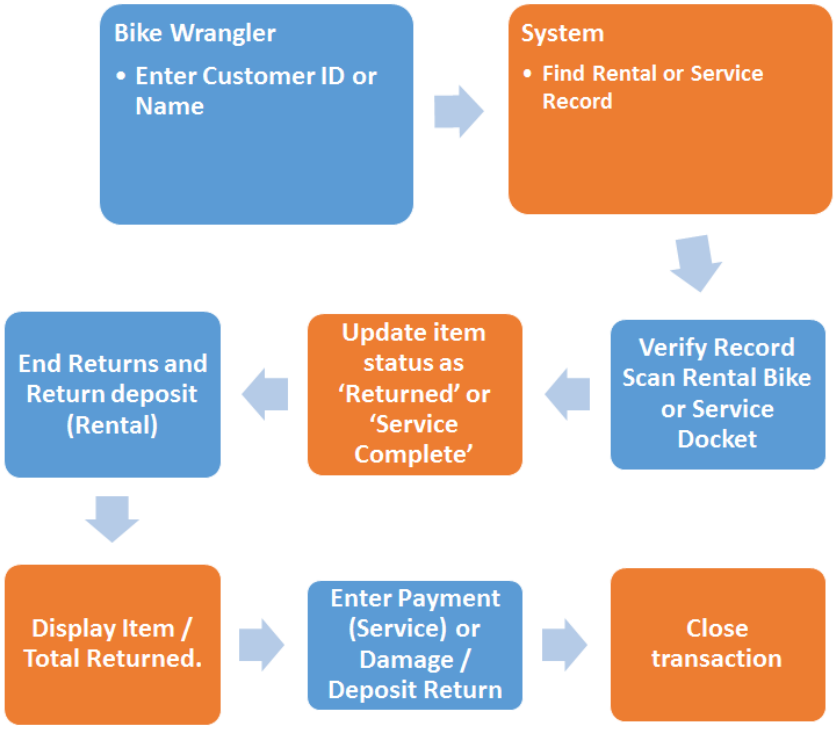
Each **use case** has been documented with a combination of activity diagrams system sequence diagrams.

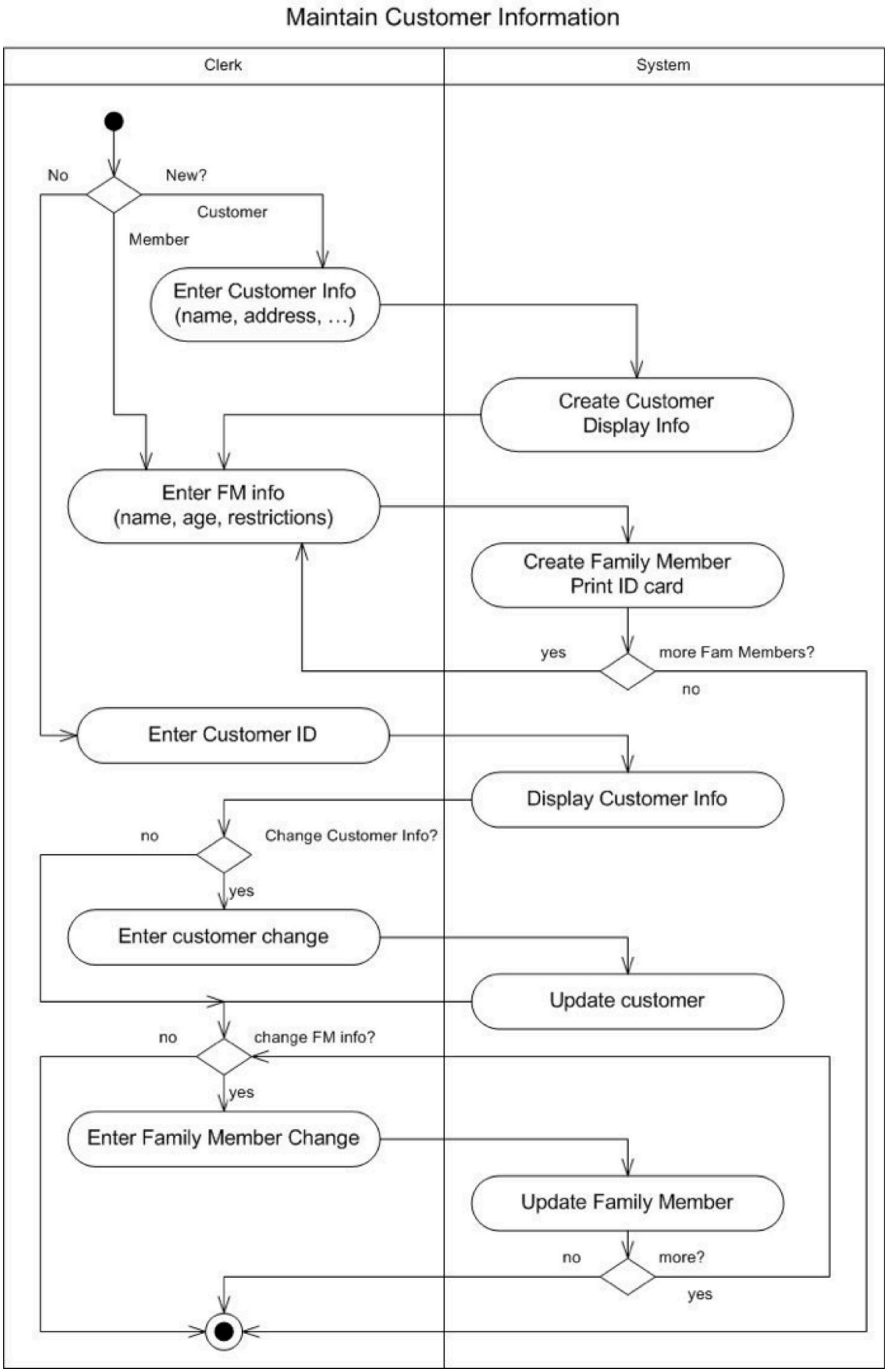
The **activity diagrams** for the use cases *check out bikes (rent bikes)*, *return bikes* and *maintain customer information* follow.

The analysts proposed that 'rentals' can be treated in the same way as other 'services' so these options are shown together.

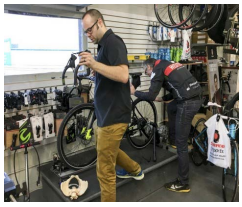
Images from http://poweredbychocolatemilk.com/blog/single/the-best-thing-you-can-do-for-your-bike-and-you/
and http://www.hartscyclery.com/


Bike Wrangler
• Enter Customer ID or Name
System
• Find Customer/Loyalty Member Display Info
Verify User / Start Rental or Other Service
Create Rental or Service Case
Check /Scan Rental or Log Bike Info for Service
Add bike info to Rental / Service Log
Finalize Rental Service
Calculate Total/Display
Enter Payment and Deposit (Rental) or Estimate (Service)
Update Rental – Print Receipt or Service Card

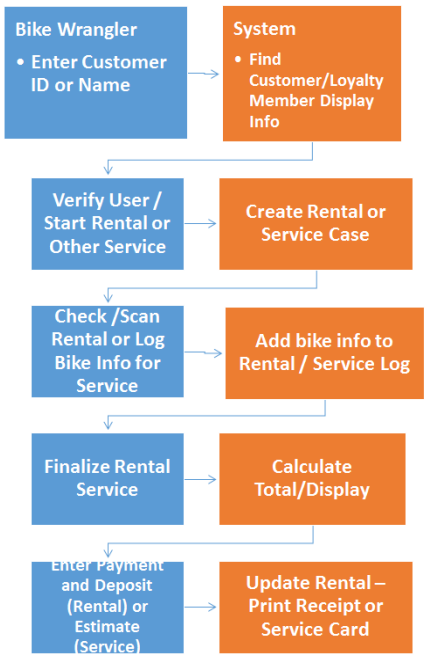

Bike Wrangler
• Enter Customer ID or Name
System
• Find Rental or Service Record
Verify Record Scan Rental Bike or Service Docket
Update item status as 'Returned' or 'Service Complete'
End Returns and Return deposit (Rental)
Display Item / Total Returned.
Enter Payment (Service) or Damage / Deposit Return
Close transaction


Maintain Customer Information
Clerk
System
No
New?
Customer
Member
Enter Customer Info
(name, address, …)
Create Customer
Display Info
Enter FM info
(name, age, restrictions)
Create Family Member
Print ID card
yes
more Fam Members?
no
Enter Customer ID
Display Customer Info
no
Change Customer Info?
yes
Enter customer change
Update customer
no
change FM info?
yes
Enter Family Member Change
Update Family Member
no
more?
yes

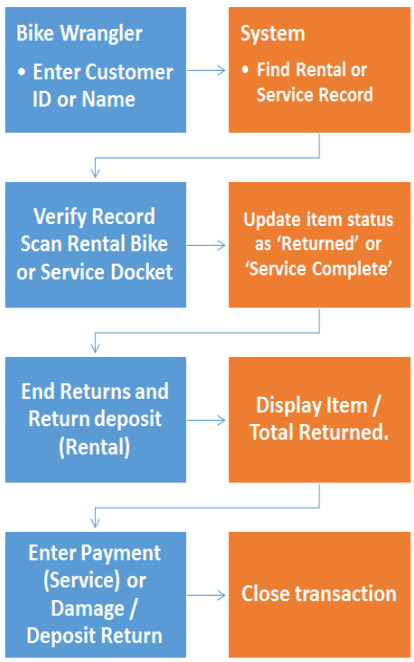
The **system sequence diagrams** for the use cases *check out bikes (rent bikes)*, *return bikes* and *maintain customer information* follow.



RENT / SERVICE BIKES



RETURN BIKES



Maintain Customer Information

A **CRUD analysis** has been conducted. This analysis is shown in the following figure. It reveals a number of additional use cases, bringing the total number of use cases to 9.

Defined Use Cases	Customer	Customer2	BikeMake	BikeType	Rental/ Service	Rental/ Service Item
Maintain Customer Information	CUD	CUD				
Enter New Bike Stock			C	C		
Rent / Service Bikes					C	
Return / CompleteS ervice						C

CRUD – Create, Read/Report, Update, Delete

Additional Use Cases	Customer	Customer2	BikeMake	BikeType	Rental/ Service	Rental/ Service Item
Customer Report	R	R				
Update Bike Stock			UD	UD		
Print Bike List			R	R		
Update Rentals /Services					UD	UD
Print outstanding rentals / Services					R	R

DownTown Bikes & Bike Rental System – Use case realization

Design the following in order, for each of the nine use cases identified during analysis.

1. A first cut sequence diagram, which only includes the relevant actor and problem domain classes (get this checked by your tutor),
2. Add the view layer classes and the data access classes to your sequence diagram
3. Develop a design class diagram based on your sequence diagram and the domain class diagram
4. Develop a package diagram showing a three-layer solution with view layer, domain layer and data access layer packages.

Use the domain class diagram, use case diagram (from the background section – above) along with the models on this page (provided by the publishers of SJB) as inputs.

When you have completed the tasks for each of the use cases integrate your domain class diagram for each use case into a single design class diagram.