

Session 1

Class and object diagrams

Class diagrams are the backbone of an object oriented design: every later diagram refers back to the classes, attributes and operations you fix here. This session draws two class diagrams and one object diagram for systems you already know as a user.

Objectives

✕ Do not copy. Read for understanding and the viva

- Complete questions 1 to 3 of the manual: class and object diagrams
- Prepare the deliverable before the lab and finish it during the session
- Be ready to explain every step in the viva

Questions Covered

✕ Do not copy. Read for understanding and the viva

Question	Requirement	Status
Q1	Draw Class Diagram for Online Shopping System (First, list the assumptions of the...	Complete
Q2	Draw Class Diagram for Online Examination System	Complete
Q3	Draw Object Diagram for Online Banking System	Complete

Preparation

✕ Do not copy. Read for understanding and the viva

- The manual asks for a problem description of 300 to 500 words and a list of assumptions before every diagram. Write both first; they fix the scope the evaluator marks you against.

- List 8 to 12 candidate classes from the nouns in your problem description, then drop the ones that are attributes.
- For each association decide multiplicity at both ends and whether it is aggregation or composition.
- An object diagram is a snapshot: pick concrete objects with real attribute values at one instant.

Question 1

Problem Statement

■ Write in lab record

Draw Class Diagram for Online Shopping System (First, list the assumptions of the shopping system and then identify classes, and then draw the diagram).

Solution

■ Write in lab record

Assumptions

The Online Shopping System is a web store run by a single retailer. A visitor registers once with a name, email, phone and password and becomes a Customer. A customer can save more than one delivery address. The catalogue is organised into categories; every product belongs to exactly one category and carries a name, a unit price and the quantity in stock. An administrator maintains the catalogue: adds products, edits prices and updates stock when a consignment arrives.

A logged-in customer browses or searches the catalogue and adds products to a shopping cart. Each cart item records one product and the quantity wanted. The cart belongs to one customer and lives as long as the account does; the customer can change quantities or remove items until checkout. At checkout the system verifies that every product is still in stock, copies the current prices into order lines and creates an Order with status Placed. The customer chooses a delivery address and a payment mode (card, UPI or cash on delivery). For card and UPI the system sends the amount to an external payment gateway; the order moves to Paid only when the gateway approves. Cash on delivery orders stay Placed until the courier collects the money.

Once an order is paid, the administrator packs it and creates a Shipment with a courier name and a tracking number. The customer tracks the shipment from the order page. A customer may cancel an order before it is shipped; the payment is then refunded through the gateway. Stock is reduced when an order is paid and restored when it is cancelled.

The system does not handle returns after delivery, seller onboarding, product reviews or discount coupons. Sales reports for the administrator are also out of scope. The diagram therefore has eleven classes: the people (Customer, Admin), the catalogue (Category, Product), the buying path (ShoppingCart, CartItem, Order, OrderLine) and the fulfilment records (Payment, Shipment, Address).

Assumptions:

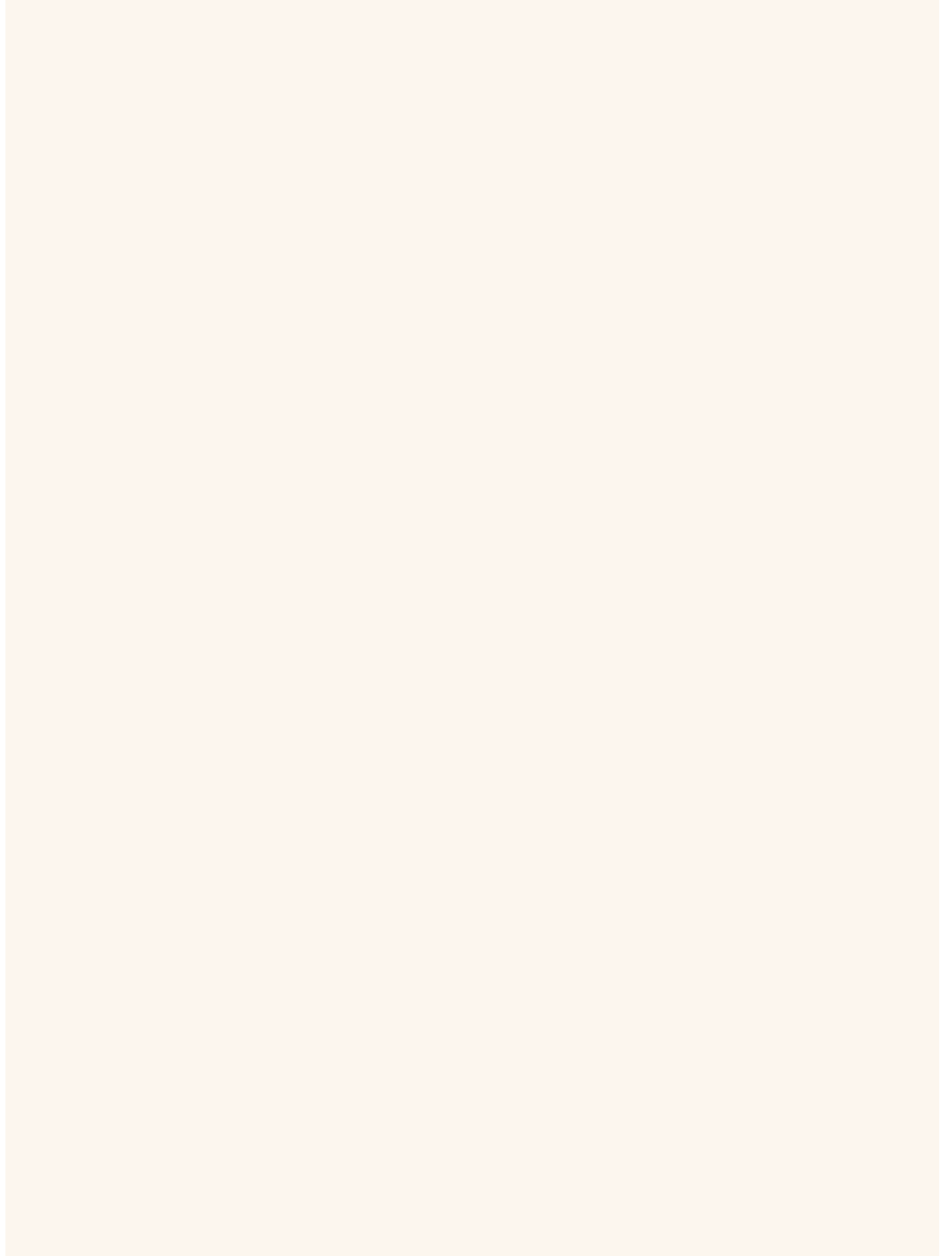
- One retailer and one warehouse; there are no marketplace sellers.
- A customer owns exactly one cart; the cart holds zero or more items and is deleted with the customer (composition).
- Prices are copied into OrderLine at checkout so a later price change does not alter an old order.
- Every order has exactly one Payment record and at most one Shipment.
- The payment gateway and the courier are external systems; they appear as attributes and operations, not as classes.
- Stock changes only when an order is paid or cancelled, never when an item is added to a cart.

Steps

1. Open StarUML, press Ctrl+N for a new project, then choose Model, Add Diagram, Class Diagram and name it `OnlineShopping`.
2. In the Toolbox click Class, then click on the canvas; type the class name. Repeat for the eleven classes in the table.
3. Select a class and press the green plus that appears on its right edge (or right-click it in Model Explorer and choose Add, Attribute) to add attributes in the form `name: type`. Add operations the same way with the Add Operation button.
4. Use Toolbox Association for plain lines, Composition for the filled diamond, Aggregation for the hollow diamond and Directed Association for the arrows to Product. Click the source class, drag to the target.
5. Select each association and, in the Editor pane on the right, set End1 and End2 multiplicity, and the name (for example `places`).

6. Save with Ctrl+S and export with File, Export, PNG for the lab record.

Diagram



Class	Attributes	Operations	Relationships
Customer	customerId: int, name: String, email: String, phone: String	register(), login(), placeOrder()	1 to 1..* Address; composes 1 ShoppingCart; places 0..* Order
Address	line1: String, city: String, pin: String	none	belongs to 1 Customer
ShoppingCart	cartId: int, createdOn: Date	addItem(), removeItem(), getTotal(), checkout()	composes 0..* CartItem
CartItem	quantity: int	subtotal()	refers to 1 Product
Category	categoryId: int, name: String	none	aggregates 0..* Product
Product	productId: int, name: String, price: double, stock: int	isAvailable(), reduceStock(), restoreStock()	in 1 Category; managed by Admin
Admin	adminId: int, name: String	addProduct(), updateStock(), shipOrder()	manages 0..* Product
Order	orderId: int, orderDate: Date, status: String, total: double	confirm(), cancel(), track()	composes 1..* OrderLine; has 1 Payment; has 0..1 Shipment; delivered to 1 Address
OrderLine	quantity: int, unitPrice: double	subtotal()	refers to 1 Product
Payment	paymentId: int, amount: double, mode: String, status: String	process(), refund()	for 1 Order

Class	Attributes	Operations	Relationships
Shipment	shipmentId: int, courier: String, trackingNo: String, status: String	dispatch(), deliver()	for 1 Order

■ Lab record: every tab is one file of the answer. Write all of them.

Sketch

```

1  +-----+ 1      1..* +-----+      +-----+ 1      0..* +-----+
2  | Customer |-----| Address |      | Category |◊-----| Product |
3  +-----+      +-----+      +-----+      +-----+
4  |1          |1                                ^ 1      ^ 1
5  |          | (comp)                                |      |
6  |          v 1                                |      |
7  |  +-----+ 1      0..* +-----+ 0..*      |      |
8  |  | ShoppingCart |<*>-----| CartItem |-----+      |
9  |  +-----+      +-----+      |      |
10 | places                                |      |
11 | v 0..*                                |      |
12 +-----+ 1      1..* +-----+ 0..*      |      |
13 | Order |<*>-----| OrderLine |-----+
14 +-----+      +-----+
15 |1          |1          |0..*
16 |1          |0..1      |1 delivered to (same Address class as above)
17 +-----+ +-----+ +-----+
18 | Payment | | Shipment | | Address |
19 +-----+ +-----+ +-----+
20
21 <*>-- filled diamond = composition      ◊-- hollow diamond = aggregation
22 → arrow = directed association      plain line = association

```

Explanation

The nouns of the description gave fourteen candidates; three of them (price, stock, tracking number) became attributes because they have no behaviour of their own and belong to one owner. Composition is used where the part cannot outlive the whole: a CartItem without its cart or an OrderLine without its order is meaningless. Category to Product is aggregation because a product can be moved to another category without being destroyed. CartItem and OrderLine both

point at Product with a one-way arrow because the product never needs to know which carts hold it. Keeping OrderLine separate from CartItem is the design decision the viva usually probes: the cart is editable and holds live prices, the order line is frozen history. These eleven classes are the lifelines used in the Session 3 sequence diagram.

Question 2

Problem Statement

■ Write in lab record

Draw Class Diagram for Online Examination System.

Solution

■ Write in lab record

Assumptions

The Online Examination System lets a university run objective tests over the web. Three kinds of users log in: students, examiners and an administrator. The administrator creates user accounts and maps each student to a programme. An examiner is responsible for one or more subjects and prepares a question bank for each subject: every question has a text, a mark value and two to four options, exactly one of which is correct.

For each subject the examiner schedules an Exam with a title, date, start time, duration in minutes and total marks, and picks the questions from the bank. A scheduled exam is visible to every student enrolled in that subject. At the start time the student clicks Start and the system creates an Attempt that records the start time. Questions are shown one at a time; the student selects an option, which the system stores as an Answer against that question. The student may revisit a question and change the selected option until submission. A countdown timer runs on the server; when the duration expires the attempt is submitted automatically.

On submission the system evaluates the attempt by comparing every stored answer with the correct option and adds up the marks. It then creates a Result with marks obtained, percentage and a pass or fail status against the pass mark set on the exam. The examiner reviews results and publishes them, after which the student can see the result and the answer key. The examiner can also see subject-wise summaries.

Each student gets a single attempt per exam. Descriptive questions, negative marking, question randomisation per student, webcam proctoring and payment of exam fees are outside the scope of this diagram. The design has eleven classes: an abstract User with the three user kinds (Student, Examiner, Admin), the content (Subject, Exam, Question, Option) and the attempt path (Attempt, Answer, Result).

Assumptions:

- All questions are multiple choice with exactly one correct option; marks are fixed per question.
- One student makes at most one attempt per exam; an attempt has at most one result.
- An exam belongs to one subject and one examiner; a question belongs to one subject but may be reused in several exams.
- Student, Examiner and Admin share login behaviour, so they inherit from an abstract User class.
- The timer is enforced by the server, so the Exam class holds the duration and the Attempt class holds the start time.

Steps

1. Model, Add Diagram, Class Diagram; name it `OnlineExamination`.
2. Add the abstract class User first: select it, tick isAbstract in the Editor pane. Add Student, Examiner and Admin and connect each to User with Toolbox Generalization (child to parent).
3. Add Subject, Exam, Question, Option, Attempt, Answer and Result with the attributes and operations from the table.
4. Draw Exam to Question as a plain association (many to many, because a question can be reused); draw Question to Option and Attempt to Answer as Composition.
5. Set multiplicities in the Editor for every association end; name the association between Student and Attempt `makes`.
6. Save and export as PNG.

Diagram

Class	Attributes	Operations	Relationships
User (abstract)	userId: int, name: String, email: String, password: String	login(), logout()	parent of Student, Examiner, Admin
Student	rollNo: String, programme: String	viewExams(), startExam(), viewResult()	enrolled in 1..* Subject; makes 0..* Attempt
Examiner	employeeId: String	addQuestion(), scheduleExam(), publishResult()	handles 1..* Subject; sets 0..* Exam
Admin	none extra	createUser(), mapStudent()	creates 0..* User
Subject	subjectCode: String, title: String	none	has 0..* Question; has 0..* Exam
Exam	examId: int, title: String, date: Date, startTime: Time, duration: int, totalMarks: int, passMarks: int	schedule(), start(), end(), isOpen()	uses 1..* Question; for 1 Subject; has 0..* Attempt
Question	questionId: int, text: String, marks: int	isCorrect(o: Option)	composes 2..4 Option; in 1 Subject
Option	optionId: int, text: String, isCorrect: boolean	none	part of 1 Question
Attempt	attemptId: int, startTime: Time, endTime: Time, status: String	begin(), answer(), submit(), evaluate()	for 1 Exam and 1 Student; composes 0..* Answer; produces 0..1 Result
Answer	answerId: int	none	for 1 Question; selects 1 Option

Class	Attributes	Operations	Relationships
Result	resultId: int, marksObtained: int, percentage: double, status: String, published: boolean	publish()	of 1 Attempt

■ Lab record: every tab is one file of the answer. Write all of them.

Sketch

```

1          +-----+ (abstract)
2          |  User  |<-----creates 0..*---+
3          +-----+
4          ^    ^    ^
5          +-----+ | +-----+
6  +-----+ +-----+ +-----+
7  | Student | | Examiner | | Admin |
8  +-----+ +-----+ +-----+
9  |1..* |1 |1 |1
10 | | |1..* |sets 0..*
11 | | +-----+ |
12 | +-----| Subject |-----+
13 | enrolled +-----+ |
14 | 1..* |1 | 0..* |
15 | 0..* | v +-----+ 1 0..* +-----+
16 | +-----+ 1..* 0..* | |-----| Attempt |
17 | | Question |-----| Exam | +-----+
18 | +-----+ uses +-----+ ^ |1 |1
19 | |1 ^ 1 for makes 0..* | | |0..
20 | 2..4 | | | | v
21 | +-----+ | +-----+ 0..* (comp) | | +-----+
22 | | Option |←+-----| Answer |-----|-----+ | Result
23 | +-----+ 1 +-----+ | +-----+
24 | selects |
25 +-----+

```

Explanation

The generalisation from User is the one inheritance in the diagram; it removes three copies of the login attributes and lets the administrator create any kind of user through one association. Exam

to Question is many to many because the bank is reused across exams; if you make it composition you cannot reuse a question. Question to Option is composition with multiplicity 2..4, which encodes the rule "at least two choices" directly in the diagram. Answer is a separate class rather than an attribute of Attempt because each answer must remember both the question and the chosen option. Result is kept apart from Attempt so that it can carry the published flag and be created only after evaluation. These classes are the lifelines of the Session 3 sequence diagram for Take Exam.

Question 3

Problem Statement

■ Write in lab record

Draw Object Diagram for Online Banking System.

Solution

■ Write in lab record

Assumptions

The Online Banking System is the net banking portal of one bank. A customer who holds at least one account at a branch applies for net banking and receives a user id and password, stored as a Credential. After logging in, the customer can view balances, see the recent transactions of any of their accounts, add a beneficiary (another account, in the same bank or in another bank, identified by account number and IFSC) and transfer funds from one of their own accounts to a saved beneficiary. Each transfer produces two Transaction records: a debit on the source account and a credit on the destination account, both carrying the same reference number. A transfer is refused when the source balance is below the amount, or when the daily transfer limit on the credential is exceeded. Cheque book requests, fixed deposits, loans, card management and bill payment are out of scope.

The class model behind the snapshot has seven classes. Bank has a name and IFSC prefix and owns many Branch objects. Branch has a branch code and city. Customer has a customer id, name and phone and belongs to one branch. Account is abstract with account number, balance and opening date, specialised as SavingsAccount (interest rate, minimum balance) and CurrentAccount (overdraft limit); a customer holds one or more accounts. Credential has a user

id, a hashed password and a daily limit and belongs to one customer. Beneficiary has a nickname, account number and IFSC, is owned by one customer and refers to the target account. Transaction has a transaction id, type (DEBIT or CREDIT), amount, date and reference, and belongs to exactly one account.

An object diagram is an instance of that class diagram at one moment. The moment chosen here is just after customer Ravi has transferred five thousand rupees from his savings account to Priya's savings account on 26 September 2026. Object names are written as `name : Class` and underlined when drawn by hand. Each attribute shows its actual value. Links are instances of the associations in the class model and are drawn as plain lines without multiplicity, because every link connects exactly two objects.

Assumptions:

- Both customers bank with the same bank so the credit side can be shown as an object; a transfer to another bank would show only the debit.
- Balances shown are the values after the transfer has been posted.
- Passwords are stored as hashes; the diagram shows a placeholder, never the real value.
- Only one branch and one beneficiary are shown to keep the snapshot readable; the class model allows many of each.
- Transaction ids are generated by the bank; the two legs of a transfer share one reference number.

Steps

1. Model, Add Diagram, Class Diagram; name it `OnlineBankingClasses` and draw the seven classes and their associations from the table (needed so the objects have classifiers).
2. Model, Add Diagram, Object Diagram; name it `OnlineBankingSnapshot`.
3. In the Toolbox click Object and click on the canvas. In the Editor pane set name to `ravi` and classifier to `Customer` (choose from the model list). Repeat for every object in the table.
4. Select each object, press the Add Slot button (or right-click in Model Explorer, Add, Slot) and set the slot name and value, for example `balance` and `40000.00`.
5. Draw links with Toolbox Link from one object to the other. Links have no multiplicity.
6. Save and export as PNG. Hand-draw the object names underlined.

Diagram

Objects at the snapshot instant:

Object	Class	Attribute values	Linked to
sbi	Bank	name = State Bank of India, ifscPrefix = SBIN	noida
noida	Branch	branchCode = 01234, city = Noida	sbi, ravi, priya
ravi	Customer	customerId = C1001, name = Ravi Kumar, phone = 9876543210	noida, login1, acc1, acc2, ben1
priya	Customer	customerId = C1002, name = Priya Singh, phone = 9123456780	noida, acc3
login1	Credential	userId = ravi1001, passwordHash = (hashed), dailyLimit = 50000.00	ravi
acc1	SavingsAccount	accountNo = SB1001, balance = 40000.00, openedOn = 2021-04-01, interestRate = 3.5, minBalance = 1000.00	ravi, t1
acc2	CurrentAccount	accountNo = CA2001, balance = 125000.00, openedOn = 2023-01-15, overdraftLimit = 50000.00	ravi
acc3	SavingsAccount	accountNo = SB1002, balance = 27000.00, openedOn = 2022-08-10, interestRate = 3.5, minBalance = 1000.00	priya, ben1, t2
ben1	Beneficiary	nickname = Priya, accountNo = SB1002, ifsc = SBIN0001234	ravi, acc3
t1	Transaction	txnId = T5001, type = DEBIT, amount = 5000.00, date = 2026-09-26, reference = TR26092601	acc1
t2	Transaction	txnId = T5002, type = CREDIT, amount = 5000.00, date = 2026-09-26, reference = TR26092601	acc3

■ Lab record: every tab is one file of the answer. Write all of them.

Sketch

```

1  +-----+ +-----+
2  | sbi : Bank      |-----| noida : Branch      |
3  | name = SBI      |      | branchCode = 01234    |
4  +-----+ +-----+
5                      |      |
6  +-----+ +-----+ +-----+
7  | login1 : Credential |--| ravi : Customer      | | priya : Customer
8  | userId = ravi1001   | | customerId = C1001    | | customerId = C1002
9  | dailyLimit = 50000.00 | | name = Ravi Kumar    | | name = Priya Singh
10 +-----+ +---+-----+ +-----+
11                      |      |      |
12 +-----+ |      | +-----+ +-----+
13 | acc1 : SavingsAccount |-----+ | ben1 : Beneficiary | |
14 | accountNo = SB1001    |      | | nickname = Priya    | |
15 | balance = 40000.00    |      | | accountNo = SB1002   | |
16 +-----+ +-----+ | +-----+ +-----+
17          |      |      |      |
18 +-----+ +-----+ +-----+ +-----+ | +-----+ +-----+
19 | t1 : Transaction      | | acc2 : CurrentAcc | +--| acc3 : SavingsAccou
20 | txnId = T5001         | | accountNo = CA2001| | accountNo = SB1002
21 | type = DEBIT          | | balance=125000.00 | | balance = 27000.00
22 | amount = 5000.00      | +-----+ +-----+
23 | reference = TR26092601 |      |
24 +-----+ +-----+
25                      | t2 : Transaction
26 (object names are underlined when drawn by hand;
27   full attribute values are in the table above) | type = CREDIT
28                      | amount = 5000.00
29                      | reference = TR26092
30                      +-----+

```

Explanation

The object diagram proves that the class model can hold a real situation. Every object maps to one class from the description; acc1 and acc3 are instances of the SavingsAccount subclass and acc2 of CurrentAccount, which is why they show different extra attributes. The two Transaction objects share the reference TR26092601 and carry opposite types, which is the check an examiner makes to confirm the transfer is modelled correctly. The balances are consistent with the story: acc1 was 45000 before the debit and acc3 was 22000 before the credit. The link from

ben1 to acc3 is what lets the Session 4 collaboration diagram route the credit to the correct account, and the link from ravi to login1 is where the daily limit is checked. Note that the diagram carries no multiplicities and no operations; both belong to the class diagram.

Viva Questions

✗ Do not copy. Read for understanding and the viva

Q: What is the difference between aggregation and composition? **A:** Both are whole-part associations. In composition the part cannot exist without the whole and is deleted with it (Order and OrderLine). In aggregation the part can be shared or outlive the whole (Category and Product).

Q: Why is OrderLine a separate class from CartItem? **A:** The cart is editable and shows the current price; the order line freezes the quantity and price at checkout so historical orders do not change when the catalogue changes.

Q: Where do the classes in a class diagram come from? **A:** From the nouns in the problem description. Nouns that have no behaviour and belong to one owner (price, tracking number) become attributes instead of classes.

Q: What does multiplicity 2..4 on Question to Option mean? **A:** Every Question object owns at least two and at most four Option objects. It encodes a business rule directly in the model.

Q: Why is User abstract in the examination diagram? **A:** Nobody is just a user; every account is a student, an examiner or an administrator. Marking User abstract stops the tool from creating a plain User object while still sharing login behaviour.

Q: How does an object diagram differ from a class diagram? **A:** An object diagram shows instances with actual attribute values and links at one moment; it has no multiplicities, no operations and no inheritance arrows. A class diagram shows the types and the rules for all moments.

Q: Why do the two transactions in the banking snapshot share one reference number? **A:** A transfer is one business event with two accounting legs, a debit and a credit. The shared reference lets the bank reverse or trace both legs together.

Q: When would you use a directed association? **A:** When only one side needs to navigate to the other. OrderLine knows its Product but a Product never needs the list of order lines that

mention it.

Common Mistakes

✖ Do not copy. Read for understanding and the viva

- Skipping the problem description and assumptions: the manual awards marks for them and they fix the scope the evaluator compares your classes against.
- Making everything a class, including price, address line and status; ask whether the noun has its own behaviour or identity.
- Drawing composition where the part is shared, for example Question composed inside Exam when the question bank is reused.
- Leaving multiplicity off one end of an association; every end needs a value, even when it is 1.
- Writing multiplicities and operations on an object diagram; it shows values and links only.
- Naming objects without a class (`ravi` instead of `ravi : Customer`) or forgetting to underline the name when drawing by hand.

Session Summary

■ Write in lab record

- Question 1: Online Shopping class diagram, eleven classes with attributes, operations and multiplicities, plus a 330 word description and six assumptions
- Question 2: Online Examination class diagram, eleven classes including an abstract User with three subclasses
- Question 3: Online Banking object diagram, eleven objects with attribute values and eleven links, taken just after a fund transfer
- Element table, ASCII sketch and a PlantUML file for every diagram, ready to hand-copy or render

[✎ Edit this page](#)