

In-Class Midterm: Briefing and Sample Paper

What is covered, how it is organised, and a worked sample

In class

Covers Weeks 1-5

Closed book, pen and paper

Scope

Weeks 1-5 (Labs 1-4): classes and objects, encapsulation, inheritance, polymorphism and binding, abstract classes and interfaces, exception handling. Learning outcomes covered: **L01** (describe OOP principles), **L02** (advantages of OO) and part of **L03** (design, develop, debug).

Format (suggested)

Part	Style	Marks
A	Short answers and true/false with a reason	20
B	Read the code: predict the output, find the compile error	30
C	Write code: a class (or two) from a description	30
D	Design: draw a UML diagram and justify one design decision	20

You will not be asked to remember library method names beyond the ones used in the labs (`ArrayList`, `System.out.println`, `Collections.sort`).

Marking approach

Part B and C marks go for **correct reasoning first**, syntax second. A missing semicolon will not cost marks. A wrong access modifier or a missing `super(...)` will.

Sample paper

Part A: short answers

1. State two advantages of OOP over procedural programming. (4)
2. Why are fields normally `private`? Give one example of a bad state that `public` fields allow. (4)
3. True or false, with a reason: "A class can implement several interfaces but extend only one class." (4)
4. What is the difference between a checked and an unchecked exception? Give one example of each. (4)
5. What does `@Override` do and why is it useful? (4)

Part B: reading code

```
abstract class Item {
    private String title;
    Item(String title) { this.title = title; }
    String getTitle() { return title; }
    abstract double fee(int days);
}
class Book extends Item {
    Book(String t) { super(t); }
    double fee(int days) { return days * 10.0; }
}
class Dvd extends Item {
    Dvd(String t) { super(t); }
    double fee(int days) { return days * 25.0; }
    String format() { return "DVD"; }
}
public class Test {
    public static void main(String[] args) {
        Item[] items = { new Book("A"), new Dvd("B") };
        for (Item i : items) System.out.println(i.getTitle() + " " + i.fee(2));
        Item x = new Dvd("C");
        System.out.println(x.format());
    }
}
```

6. What does the loop print? (5)
7. The last line does not compile. What is the error, why does it happen, and give **two** ways to fix it. (8)
8. Add a class `Magazine` extends `Item` that omits `fee`. What error does the compiler give? (4)
9. What is the *declared* type and the *real* type of `x`? (3)

Part C: writing code

10. Write a class `Member` with private `id` and `name`, a constructor that uses its setters, and setters that throw `IllegalArgumentException` for blank values. (12)
11. Write a checked exception `LimitReachedException` and a method `borrow()` in `Member` that throws it when `count >= limit`. Show how a caller handles it. (10)
12. Write an interface `Reservable` and make `Book` implement it. (8)

Part D: design

13. A library adds `AudioBook`: it is loanable, reservable, and has a duration. Draw the UML (name, one field, key methods) showing where it sits. (10)
14. Argue for **either** an abstract class **or** an interface for `Loanable`. (10)

Model answers (outline)

Q	Key points
1	Any two of: modularity, reuse, maintainability, real-world modelling, teamwork
2	Only the class's code can change the data; validation lives in one place. Example: <code>title = ""</code> , negative count
3	True: Java has single inheritance of classes but multiple interface implementation
4	Checked: compiler forces handling, <code>extends Exception</code> (e.g. <code>ItemNotAvailableException</code>). Unchecked: <code>RuntimeException</code> (e.g. <code>IllegalArgumentException</code>)
5	Tells the compiler "this overrides a parent method"; catches misspelt names and wrong parameters
6	A 20.0 then B 50.0
7	<code>x</code> is declared <code>Item</code> ; <code>format()</code> exists only in <code>Dvd</code> : <i>cannot find symbol</i> . Fix: declare <code>Dvd x = new Dvd("C");</code> or cast after <code>instanceof</code> (<code>if (x instanceof Dvd d) d.format()</code>)
8	<code>Magazine</code> is not abstract and does not override abstract method <code>fee(int)</code> in <code>Item</code>
9	Declared <code>Item</code> , real <code>Dvd</code>
14	Interface is natural: a <i>capability</i> shared by unrelated classes, and items extend only one parent

How to prepare

- Redo the Week 6 rebuild challenge under time pressure.
- Re-read your own PR descriptions: the "what was hard" sections show you where to revise.
- Explain each concept aloud to a classmate in one minute. If you cannot, revise it.