1. OOP មានអត្ថប្រយោជន៍ជាច្រើន៖

Modularity – កូដត្រូវបានបំបែកជាថ្នាក់ (class) ងាយថែទាំ

Reusability – ការប្រើការបស (inheritance) & polymorphism

Encapsulation – ការការពារទិន្នន័យក្នុង class

Abstraction – លាក់លូចភាពស្មុគស្មាញ

Maintenance & Scalability – កម្មវិធីធំនឹងងាយអភិវឌ្ឍបន្ថែម

2. ខុសគ្នារវាង Class និង Object

Class	Object
គ្រោង/គំរូ (Blueprint)	វត្ថុពិត (Instance)
មិនកាន់ Memory	កាន់ Memory
កំណត់ Property និង Behavior	មាន state ពិតប្រាកដ
ឧទាហរណ៍: Car	new Car( )

3. ខុសគ្នារវាង Compile-time និង Run-time Polymorphism

Compile-time polymorphism

- កំណត់នៅពេល **compile**
- ប្រើ **method overloading**
- ល្បឿនល្អ (fast)

Run-time polymorphism

- កំណត់នៅពេល **program run**
- ប្រើ **method overriding**
- ជាប់ dynamic binding (ស្មើរវែងជាង)

4. ស្រដៀង & ខុសគ្នារវាង Static Class និង Abstract Class

ស្រដៀងគ្នា

- មិនអាចបង្កើត object ដោយផ្ទាល់
- មាន field និង method
- សូមប្រើក្នុងការរៀបចំ logic

Static Class	Abstract Class
ទាំងអស់ត្រូវ static	មាន method ធម្មតា & abstract
មិនអនុញ្ញាតឲ្យស	អនុញ្ញាត
សម្រាប់ utility/helper	សម្រាប់ Base class

Static Class	Abstract Class
មិនអាច override	អាច override abstract method

5. ខុសគ្នារវាង Method Overloading និង Method Overriding

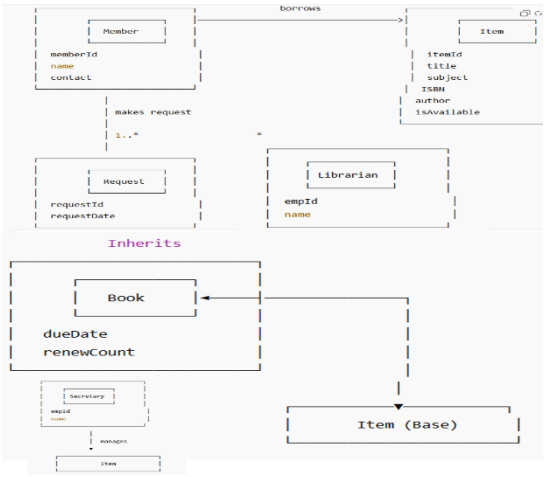
Overloading

- Same name, different parameters
- Compile-time polymorphism
- ក្នុង class ដូចគ្នា

Overriding

- Same name, same parameters
- Run-time polymorphism
- នៅក្នុង subclass

6. A. Class Diagram (Text-Based)



b. Explain

(Classes)

- **Member (សមាជិក):** តំណាងឲ្យអ្នកដែលបានចុះឈ្មោះ ដែលអាច **ខ្ចី (borrows)** ឯកសារ និង **ដាក់សំណើ (makes request)**។
  - *គុណលក្ខណៈ:* memberId, name, contact។
- **Item (ឯកសារ):** តំណាងឲ្យទ្រព្យសម្បត្តិទូទៅរបស់បណ្ណាល័យ (ដូចជាសៀវភៅ ឬទស្សនាវដ្តី)។
  - *គុណលក្ខណៈ:* itemId, title, subject, ISBN, author, isAvailable (សម្រាប់ពិនិត្យមើលថាឯកសារនោះទំនេរឬអត់)។
- **Request (សំណើ):** តំណាងឲ្យការស្នើសុំឯកសារណាមួយដោយសមាជិក។
  - *គុណលក្ខណៈ:* requestId, requestDate។
- **Book (សៀវភៅ):** ជាប្រភេទពិសេសរបស់ **Item (ឯកសារ)** (ទំនាក់ទំនងតំណពូជ **Inherits** / **Generalization** / **បន្តវេន** ឬ **Item (Base)** ជាថ្នាក់មូលដ្ឋាន)។
  - *គុណលក្ខណៈ:* ក្រៅពីគុណលក្ខណៈរបស់ Item គឺមានបន្ថែម dueDate (កាលបរិច្ឆេទកំណត់សង) និង renewCount (ចំនួនដងដែលបានបន្ត)។

- **Librarian (បណ្ណារក្ស)** និង **Secretary (លេខា):** តំណាងឲ្យបុគ្គលិកបណ្ណាល័យ។

២. ទំនាក់ទំនងសំខាន់ៗ

- **Member borrows Item (សមាជិកខ្ចីឯកសារ):** គឺជាទំនាក់ទំនងប្រើប្រាស់ (Association)។
- **Member makes request (សមាជិកដាក់សំណើ):** សមាជិកម្នាក់អាចដាក់សំណើ **មួយ ឬច្រើន (1..*)**។
- **Secretary manages Item (លេខាគ្រប់គ្រងឯកសារ):** គឺជាទំនាក់ទំនងប្រើប្រាស់។
- **Book Inherits Item (សៀវភៅបន្តរាងពីឯកសារ):** មានន័យថាសៀវភៅគឺជាឯកសារមួយប្រភេទ។ ថ្នាក់ Book ទទួលមរតកនូវគុណលក្ខណៈទាំងអស់ពីថ្នាក់ Item។

c. Relationship

Relationship	Meaning
Member → Item (Borrow)	Association
Member → Request	One-to-many
Item → Book	Inheritance
Librarian → Request	Association
Secretary → Item	Association (inventory control)
Member → Item (Renew)	Conditional association

d. Skeletal Code

```
public class Item {
    protected String itemId;
    protected String title;
    protected String subject;
    protected String isbn;
    protected String author;
    protected boolean isAvailable = true;

    public void borrow() {
        isAvailable = false;
    }

    public void returnItem() {
        isAvailable = true;
    }

    public boolean canRenew() {
        return isAvailable;
    }
}
```

```
public class Book extends Item {
    private Date dueDate;
    private int renewCount;

    public void setDueDate(Date due) {
        this.dueDate = due;
    }

    public void renew() {
        if (canRenew()) {
            renewCount++;
            // Extend date here...
        }
    }
}
```

```
public class Secretary {
    private String empId;
    private String name;

    public void addItem(Item item) {
        // Add item to library
    }

    public void removeItem(Item item) {
        // Remove item from library
    }
}
```

```
public class Member {
    private String memberId;
    private String name;
    private List<Item> borrowedItems;

    public void borrowItem(Item item) {
        borrowedItems.add(item);
        item.borrow();
    }

    public Request makeRequest(Item item) {
        return new Request(this, item);
    }
}
```

```
public class Request {
    private String requestId;
    private Member member;
    private Item item;
    private Date requestDate;

    public Request(Member m, Item i) {
        this.member = m;
        this.item = i;
        this.requestDate = new Date();
    }
}
```

```
public class Librarian {
    private String empId;
    private String name;

    public void processRequest(Request request) {
        // Approve or reject request
    }
}
```