โครงงาน

Numerical Method
Calculator Website

จัดทำโดย

6504062620175 นายอธิชา เล็กสรรเสริญ

เสนอ

ผู้ช่วยศาสตราจารย์สถิต ประสมพันธ์

วิชา 040613204 Object-Oriented Programming

ภาคเรียนที่ 1 ปีการศึกษา 2566

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

บทที่ 1 **ที่มาและความสำคัญของโครงการ**

โครงงานนี้จัดขึ้นเพื่อวัดผลความสามารถในการเรียนวิชา Object Oriented Programming โดยการนำเรื่องที่เรียนมาสร้างเป็นชิ้นงานในรูปแบบเว็บ โดยใช้แนวคิดการเขียนโปรแกรมแบบเชิงวัตถุ และยังช่วยให้ผู้จัดทำเรียนรู้อุปกรณ์และเครื่องมือ ผู้จัดทำได้สร้างเว็บนี้ขึ้นมา

ประเภทของโครงการ

โปรแกรมเว็บแอปพลิเคชัน Full-Stack

ประโยชน์

1. เพื่อให้สามารถคำนวนปัญหาทาง Numerical ได้อย่างสะดวกสบาย
2. เพื่อนำความรู้จากวิชา Database, Numerical Methods, Object-Oriented Programming มาประยุกต์ใช้
3. เพื่อนำแนวคิดการเขียนโปรแกรมแบบเชิงวัตถุมาประยุกต์ใช้

ขอบเขตของโครงการ

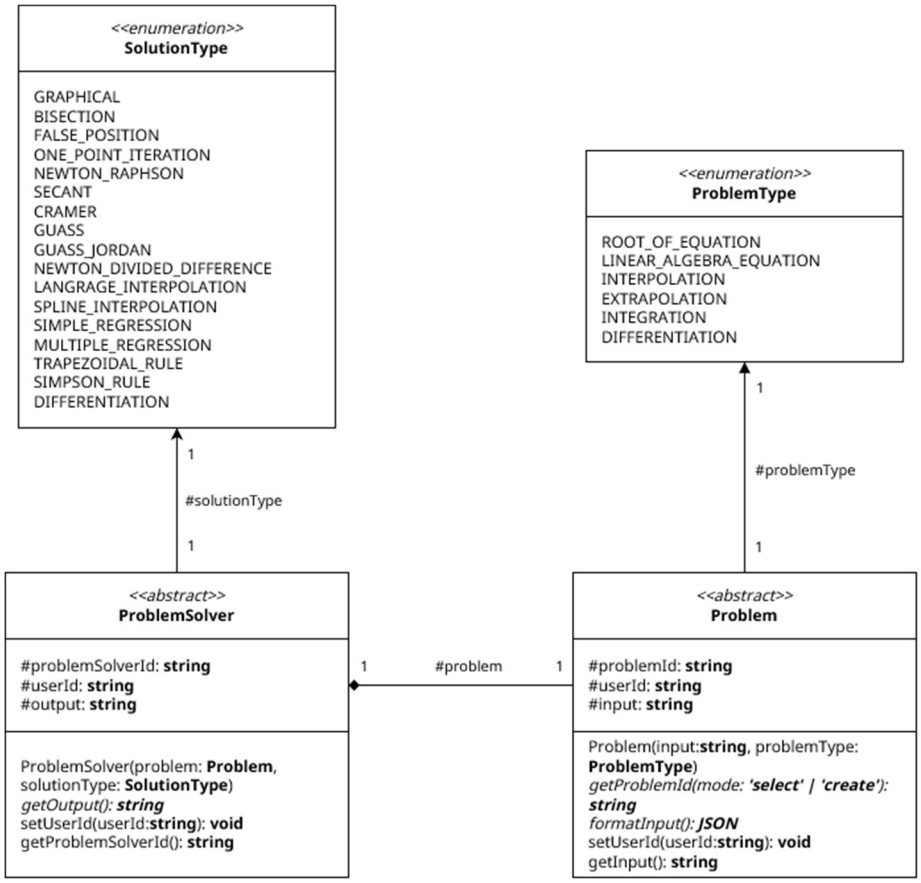
1. ความต้องการของระบบ (Functional Requirements)
   - สามารถคำนวนปัญหาทาง Numerical ดังนี้ได้
     - รากของสมการ (Root of Equation)
     - สมการพีชคณิตเชิงเส้น (Linear Algebra Equation)
     - การประมาณค่าในช่วง (Interpolation)
     - การประมาณค่านอกช่วง (Extrapolation)
     - ปริพันธ์ (Integration)
     - อนุพันธ์ (Differentiation)
   - สามารถแสดงผลลัพธ์ออกเป็นกราฟแบบโต้ตอบ (Interactive graph) โดยใช้ ploly.js
   - สามารถเก็บข้อมูลของปัญหาที่เคยคำนวณ โดยใช้ PlanetScale Database
   - สามารถเข้าสู่ระบบด้วย Google Account
   - สามารถแสดงปัญหาที่เคยคำนวนแล้ว และแสดงจำนวนครั้งที่ถูกคำนวณได้

2. ตารางเวลาการดำเนินโครงการ (Project Schedule)

งาน	สัปดาห์ 1 (ก.ย.)			สัปดาห์ 2 (ต.ค.)			สัปดาห์ 3 (ต.ค.)			สัปดาห์ 4 (ต.ค.)			สัปดาห์ 5 (ต.ค.)		
	25	27	29	2	4	6	9	11	13	16	18	20	23	25	27
1. ติดตั้งและออกแบบฐานข้อมูล	x														
2. ระบบเข้าสู่ระบบด้วย Google		x													
3. ปัญหา Root of Equation			x	x	x	x									
4. กราฟแบบโต้ตอบ (plotly.js)							x								
5. ปัญหา Linear Algebra Equation								x	x	x					
6. ปัญหา Interpolation											x	x			
7. ปัญหา Extrapolation													x		
8. ปัญหา Integration														x	
9. ปัญหา Differentiation															x
	0%	6%	13%	22%	28%	34%	44%	50%	56%	66%	72%	78%	88%	94%	100%

บทที่ 2 **การพัฒนา**

แผนภาพ Class Diagram



โครงการนี้จะมีคราสหลักอยู่สองคราสคือ

1. คราส Problem จะเก็บข้อมูลของปัญหาต่างๆ เช่น
    - Id ของปัญหา
    - Id ของผู้ใช้งาน (ที่ถามปัญหา)
    - รายละเอียดของปัญหาซึ่งจะเก็บในรูปของ **JSON** ที่เป็น **string**
    - ประเภทของปัญหา เช่น Root of equation, ...

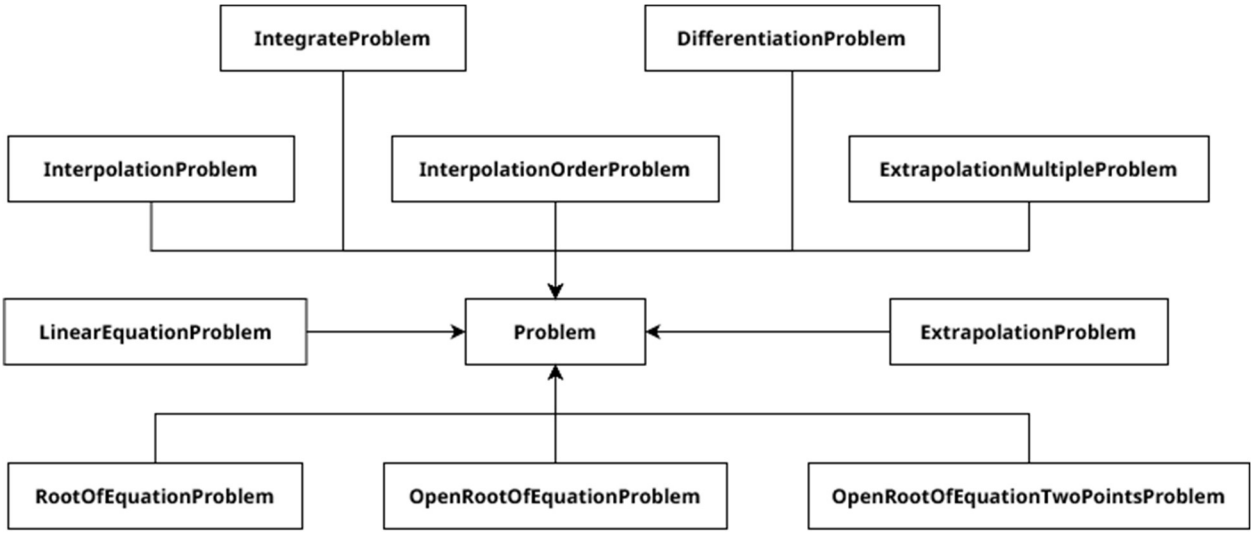
    และจะมี Methods ต่างๆ ดังนี้

    - getProblemId(mode: ‘select’ | ‘create’) ซึ่งเป็น Abstract method โดย methods นี้จะสามารถเลือกได้ว่าจะสร้าง Problem ใหม่หรือดึงจากที่มีอยู่บนฐานข้อมูล ซึ่งแต่ละปัญหาจะมีการดึงข้อมูลจากฐานข้อมูลที่แตกต่างกัน จึงสามารถ Inherit คราสนี้เพื่อไปสร้างเป็นปัญหาต่างๆ ได้
    - setUserId(userId: string) ตั้ง Id ของผู้ใช้งาน (ที่ถามปัญหา)

- getInput() ส่งรายละเอียดของปัญหา
- formatInput() เป็น Abstract class ที่จะเช็คว่าข้อมูลที่ส่งมาจากผู้ใช้งานให้ตรงกับปัญหาที่ถาม หากไม่ตรงจะไม่สามารถแก้ปัญหานั้นได้ หรือข้อมูลเยอะเกินที่ระบบรองรับ

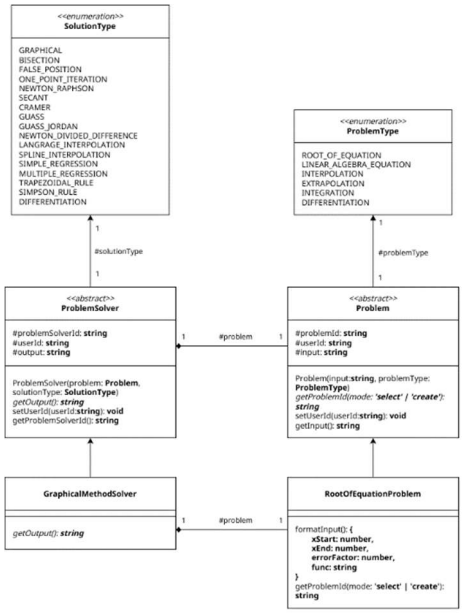
2. คราส ProblemSolver จะเก็บวิธีการแก้ปัญหารูปแบบต่างๆ ของปัญหานั้นๆ เช่น
    - Id ของ Solution
    - Id ของผู้ใช้งาน (ที่ถามปัญหาด้วย Solution นี้)
    - รายละเอียดของวิธีการแก้ปัญหาซึ่งจะเก็บอยู่ในรูปของ **JSON** ที่เป็น **string**
    - ประเภทของวิธีแก้ปัญหา เช่น Graphical Method, ...

    และจะมี Methods ต่างๆ ดังนี้
    - getOutput() เป็น Abstract Method ที่จะดึง input ของคราส Problem มาแก้ปัญหาต่างๆ ตามวิธีการแก้ปัญหาของ Object นั้นๆ โดย Methods จะดึงวิธีแก้ปัญหามาจากฐานข้อมูลและเพิ่มจำนวนการแก้ ถ้าหากเคยมีคนถามคำถามนี้แล้ว หากไม่มีจะทำการเพิ่มเข้าไปในฐานข้อมูล
    - setUserId(userId: string) ตั้ง Id ของผู้ใช้งาน (ที่ถามปัญหาด้วย Solution นี้)
    - getProblemSolverId() โดย Method นี้จะดึง Id ของ Solution จากฐานข้อมูล จาก Id ของปัญหาและวิธีการแก้ปัญหา หากมีจะตั้งให้ output เป็นคำตอบจากฐานข้อมูลเลย



แผนภาพการสืบทอดคราส ของคราส Problem

(แบบง่าย)




```ts
export class RootOfEquationProblem extends Problem {
    constructor(input: string) {
        super(input, 'ROOT_OF_EQUATION');
    }

    formatInput(): [
        null | { xStart: number; xEnd: number; errorFactor: number; func: string },
        null | { message: string; status: number }
    ] {
        // ...
    }

    async getProblemId(
        mode: 'select' | 'create' = 'select'
    ): Promise<[null | string | undefined, null | { message: string; status: number }]> {
        // ...
    }
}

export class OpenRootOfEquationProblem extends Problem {
    constructor(input: string) {
        super(input, 'ROOT_OF_EQUATION');
    }

    formatInput(): [
        null | { xStart: number; errorFactor: number; func: string },
        null | { message: string; status: number }
    ] {
        // ...
    }

    async getProblemId(
        mode: 'select' | 'create' = 'select'
    ): Promise<[null | string | undefined, null | { message: string; status: number }]> {
        // ...
    }
}
```

ตัวอย่างของการ Implement คราส Graphical Method และ
ปัญหาของ Root Of Equation

รูปแบบการพัฒนาโครงการ

- ภาษา: Svelte, TypeScript
- Framework: SvelteKit (Svelte + Vite)
- ผู้ให้บริหารโฮสติ้ง: Vercel
- ฐานข้อมูล
  - Prisma เป็นไลบรารี่จับคู่เชื่อมโยงระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ หรือ ORM
  - PlanetScale ผู้ให้บริหารฐานข้อมูลแบบ MySQL
- CSS และ Components:
  - TailwindCSS เป็น CSS Utility Framework
  - shadcn-svelte เป็นไลบรารี่ที่รวม Components ต่างๆ
  - KaTeX เป็นไลบรารี่แสดงสมการทางคณิตศาสตร์
- กราฟแบบโต้ตอบ: Plotly.js
- ไลบรารี่คำนวนทางคณิตศาสตร์: Math.js
- ไลบรารี่ระบบล็อคอิน: Lucia

แนวคิดการเขียนโปรแกรมเชิงวัตถุ

- **Constructor**

```ts
// src/lib/server/problem.ts
constructor(input: string, problemType: ProblemType) {
    this.input = input;
    this.problemType = problemType;
}
```

```ts
// src/lib/server/problem.ts
constructor(problem: Problem, solutionType: SolutionType) {
    this.problem = problem;
    this.solutionType = solutionType;
}
```

Constructor ของคราส Problem และคราส ProblemSolver เพื่อรับ Parameters ต่างๆ มาตั้ง Attribute

```ts
// src/lib/server/rootProblem.ts
constructor(input: string) {
    super(input, 'ROOT_OF_EQUATION');
}
```

เมื่อคราส RootOfEquationProblem inherit คราส Problem มาจึงไม่จำเป็นต้องใส่ solutionType แต่สามารถเรียก super class Constructor ได้เลย

- **Encapsulation**

```ts
// src/lib/server/problem.ts
protected problemSolverId?: string;
protected solutionType: SolutionType;
protected userId?: string;
protected problem: Problem;
protected output?: string;
```

Attribute ของคราสหลักๆ เช่นคราส Problem และคราส ProblemSolver จะเป็นแบบ Protected เพื่อให้สามารถมองเห็นได้เฉพาะคราสที่ Inherit ไปเช่น คราส CramerSolver, คราส GuassEliminationSolver, ... โดยหากต้องการค่าของ Attribute สามารถใช้ได้จาก getter methods เช่น getInput(), getOutput()

- **Composition**

```ts
// src/lib/server/problem.ts
protected problem: Problem;
```

ในคราสของ ProblemSolver จะมี Attribute เป็นวัตถุ (Object) ของคราส Problem ซึ่งไว้เก็บปัญหาที่คราส ProblemSolver นั้นๆ พยายามแก้ ตัวอย่างเช่นคราส TrapezoidalRuleSolver จะมี Class ปัญหาคือคราส IntegrateProblem

```ts
// src/lib/server/problem.ts
const [problemId, problemIdError] = await this.problem.getProblemId('select');
```

การเรียกใช้งานคราส Problem
ในคราส ProblemSolver

- **Polymorphism**
- **Abstract**
- **Inheritance**