SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

소프트웨어 유지보수

9기 계절학기

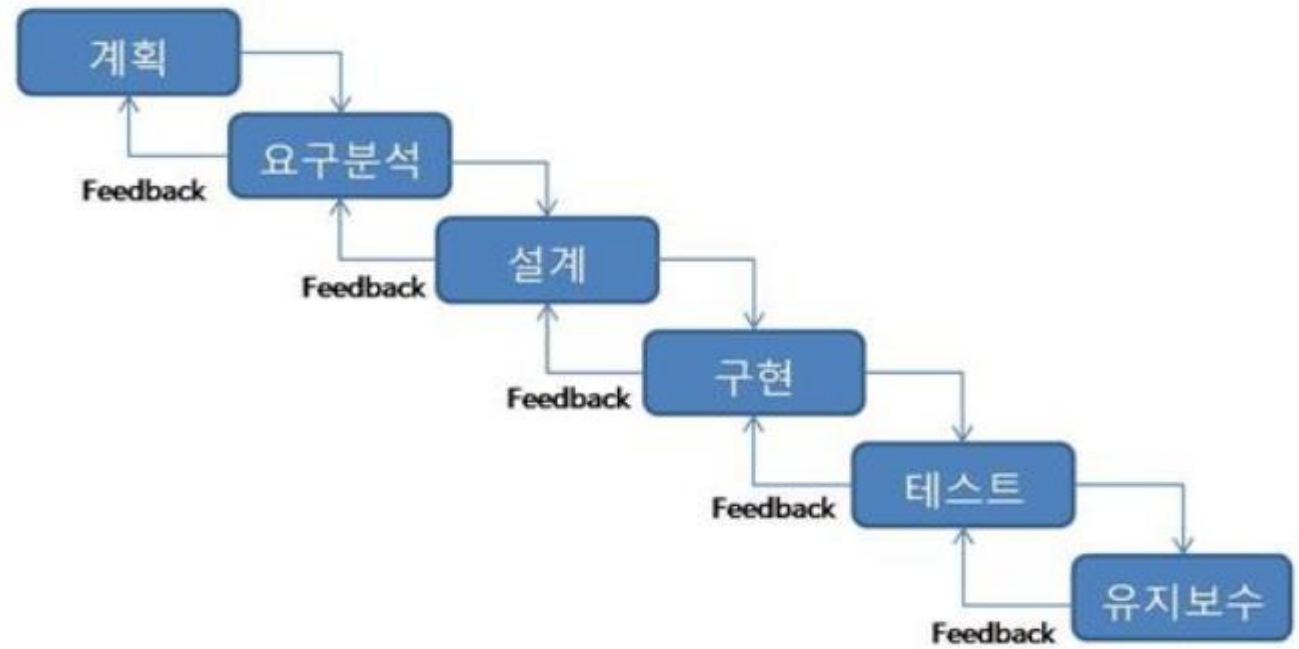
과제

- ✓ 1학기에 본인이 진행한 관통 프로젝트를 기준으로 유지보수 유형 中 Preventive Maintenance(예방 보수)의 한가지인 Refactoring을 진행해주세요
  - 본인의 소스에서 가독성이 낮은 코드나 동일 로직이 중복적용된 부분을 찾아내어 확장성 있게 수정하면 됩니다

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로! Enabling People

소프트웨어 생애주기



SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로! Enabling People

소프트웨어 유지보수란?

- ✓ 소프트웨어 유지보수는 현업의 요구에 대응하기 위해 변경하는 프로세스입니다
- ✓ 제품이 출시된 후 전반적인 개선, 오류나 버그 수정 및 성능 개선 등의 다양한 이유로 진행됩니다

소프트웨어 유지보수의 유형

- ✓ Corrective Maintenance (수정 보수)
- ✓ Adaptive Maintenance (적응 보수)
- ✓ Perfective Maintenance (향상 보수)
- ✓ Preventive Maintenance (예방 보수)

Corrective Maintenance

- ✓ 하자의 원인을 찾아 문제를 해결하는 것
- ✓ 오류를 수정하기 위해 필요한 변경 작업

Adaptive Maintenance

- ✓ 운영체제, 하드웨어와 같은 프로그램 환경변화에 맞추기 위해 수행

Perfective Maintenance

- ✓ 기존 기능과 다른 새로운 기능을 추가
- ✓ 기존 기능을 개선
- ✓ 유지보수 작업의 대부분을 차지

Preventive Maintenance

- ✓ 소프트웨어의 이해도를 높이기 위해 프로그램 구조를 변경(리팩토링)
- ✓ 잠재적 오류 발생에 대비하여 미리 예방 수단 강구

Code Refactoring

- ✓ 여러 유지보수 유형 중 Preventive Maintenance에 해당
- ✓ 기존에 동작하던 본연의 기능은 그대로 둔 채..
- ✓ 알아보기 쉽고 수정하기 간편하게 소프트웨어 내부를 수정하는 작업
- ✓ 판단 기준에는 확장성, 가독성, 유지보수성이 있습니다

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로! Enabling People

Code Refactoring

- ✓ 해당 과제에 정해진 답은 없습니다..
- ✓ 다음페이지의 예시를 참조해서 각각의 케이스에 맞게 수행하세요

Extract Function

✓ 파면화된 코드의 모듈화

```
function printOwing(invoice) {
  printBanner();
  let outstanding  = calculateOutstanding();

  //print details
  console.log(`name: ${invoice.customer}`);
  console.log(`amount: ${outstanding}`);
}
```

```
function printOwing(invoice) {
  printBanner();
  let outstanding  = calculateOutstanding();
  printDetails(outstanding);

  function printDetails(outstanding) {
    console.log(`name: ${invoice.customer}`);
    console.log(`amount: ${outstanding}`);
  }
}
```

Decompose Conditional

✓ 복잡한 if-then-else 조건문을 갖고 있다면, 각 영역을 메소드로 분리

```
if (!aDate.isBefore(plan.summerStart) && !aDate.isAfter(plan.summerEnd))
  charge = quantity * plan.summerRate;
else
  charge = quantity * plan.regularRate + plan.regularServiceCharge;
```

```
if (summer())
  charge = summerCharge();
else
  charge = regularCharge();
```

Replace Error Code with Exception

✓ 오류가 발생하면 예외처리를 사용하도록 한다

```
if (data)
  return new ShippingRules(data);
else
  return -23;
```

```
if (data)
  return new ShippingRules(data);
else
  throw new OrderProcessingError(-23);
```

Split Loop

✓ 두가지를 수행하는 루프가 있다면, 분리한다

```
let averageAge = 0;
let totalSalary = 0;
for (const p of people) {
  averageAge += p.age;
  totalSalary += p.salary;
}
averageAge = averageAge / people.length;
```

```
let totalSalary = 0;
for (const p of people) {
  totalSalary += p.salary;
}

let averageAge = 0;
for (const p of people) {
  averageAge += p.age;
}
averageAge = averageAge / people.length;
```

Preserve Whole Object

✓ 객체에서 몇 개의 값을 받아서 메소드 호출 시 파라미터로 사용중이라면 객체 전체로 보냅니다

```
const low = aRoom.daysTempRange.low;
const high = aRoom.daysTempRange.high;
if (aPlan.withinRange(low, high))
```

```
if (aPlan.withinRange(aRoom.daysTempRange))
```