시험 답안 영어로 작성?

책 단원별 문제 풀어봐야 하는지?

Access identifier 4가지의 종류별 특징?

자바의 정석?

작년에 몬테카를로 문제 어떻게 나왔는지 물어보기

1 8 23

1 11 52

* 몬테 카를로 시뮬레이션, matrix, arraylist, 성적표 프로그램

Data Structure: 데이터를 어떻게 저장할 것인가

Reference Data Type

* Array: 동일한 데이터 타입 n개를 하나의 identifier로 access하기 위해 만들어진 데이터 구조 (같은 타입의 데이터 여러 개를 한 번에 저장)
* Structure(구조체): 서로 다른 데이터 타입을 하나의 identifier로 access하기 위해 만들어진 데이터 구조
* Class: user-defined data type

Java 사용 이유: 공짜, OOP, platform independent

OOP(Encapsulation, Inheritance, Polymorphism )의 목적: 코드의 재사용성 향상과 유지보수를 위함

1. Encapsulation: 명사적 특징(attribute)와 동사적 특징(method)을 하나의 데이터 타입인 Class로 만듦. Private keyword를 사용해 information hiding(정보 은닉)함.
2. Inheritance:
3. Polymorphism: 하나의 객체가 여러 개의 자료형 타입을 가질 수 있는 것

Class(Driving Class: main method 있는 클래스, Driven Class: main method 없이 호출되는 역할)

1. Instance Methods: operate on individual objects of a class
   1. Constructor(특징 3가지 반드시 외우기)
      1. Method 이름이 class 이름과 동일함
      2. Return 타입 자체가 없음
      3. 명시적으로 call하지 않아도 object를 만드는 시점에 자동으로 실행됨

* Default constructor(automatically provided by compiler) vs No-argument constructor(explicitly provided)
  1. Accessor: attribute에 access만 하겠다
  2. Modifier: attribute을 수정하겠다.

1. Static Method: performs for the entire class, not its individual variables
2. Identifier
   1. Access identifier
      1. Public: 모든 객체에서 접근 가능(제한 없음)
      2. Private: 해당 클래스에서만 접근 가능 🡪 information hiding (정보 은닉)
      3. Protected: 해당 패키지의 클래스 혹은 해당 패키지를 상속받은 클래스에서 접근 가능
      4. Default: 해당 패키지 내에서 접근 가능
   2. Method identifier
      1. Static: object를 만들지 않고도 사용 가능하다는 의미
3. Object: 메모리를 allocation하고 메모리에 access할 수 있는 identifier까지 만들게 되면 그 자체를 object라고 함 / user-defined data type인 클래스에서 정의한 것
4. Local variable, static variable, instance variable

* Static variable, instance variable, methods SCOPE: Inside the class
* Local variable SCOPE: From the point where it is declared to the end of the block in which its declatation occurs.
* Having two references for the same object is known as **aliasing.**
* If a reference is a null and gets invoked, a **NullPointerException** will occur.

Input 방법 세가지

1. 메인 메서드 input
   1. https://javacpro.tistory.com/11
   2. Integer.parseInt(args[0])
   3. 프로그램 시작할 때 한 번만 가능(입력은 모두 String이므로 알아서 형변환 해야함)
2. API Class 사용
   1. Chn 패키지(Chnutil)
   2. Scanner, BufferedReader, Console
3. GUI를 통한 input

Output 방법

1. Print, println, printf 구분
2. Concat, +, StringBuilder

소스코드에서 dot(.)의 의미

1. Import 할 때 directory의 의미
2. Member access operator

명령 프롬프트에서 dot(.)의 의미

1. Dot 한 개는 현재 디렉토리
2. Dot 두 개는 상위 디렉토리

박홍제T가 숙제로 내주신 것 다시 해보기

==(equal to operator)와 equals() 메소드 구분

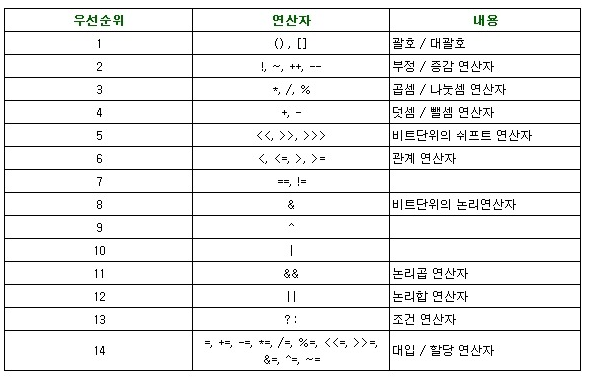
실수형의 비교

- if(Math.abs(sectionID - currentSectionID) < epsilon)

- Float.compare(a, b), Double.compare(a, b)에서 둘이 같으면 0, a가 크면 1, b가 크면 -1

Main 메소드는 static 메소드이며(따라서 객체 만들지 않아도 되므로 클래스에서 main 메소드가 가장 먼저 실행됨, driving 메소드라고도 불림)

연산자 우선 순위



1. 단항연산자
2. 사칙연산자
3. 관계연산자 – 부등호 붙은 것이 우선
4. 논리연산자 – 세 개가 우선순위가 다 다름
5. Assignment operator

* 캐스팅 연산자가 사칙연산자보다 높은 우선순위

Short-circuit property: (In java logical operator) if the evaluation of a logical expression exits is between before complete evaluation, it is called to be Short-circuit

Iteration에서 중요한 것: for-each문은 element들을 modify할 수 없다.

실 매개 변수와 형식매개 변수

* 함수에 전달되는 매개변수를 실제 매개변수(인자)라고 부르고 함수가 전달받은 매개변수를 형식 매개변수(매개변수)라고 부름

Call by value(값에 의한 호출): 매개변수의 타입이 기본 데이터 타입인 경우 🡪 실 매개변수의 값이 영향을 받지 않음(method에서 사용할 때)

* Call by reference(주소에 의한 호출): 매개변수의 타입이 참조 데이터 타입인 경우 🡪 실 매개변수의 값이 영향을 받음(method에서 사용할 때)

Error

1. Syntax Error: keyword를 잘못 사용했을 때. 컴파일러가 지속적으로 지적해주므로 비교적 찾기 쉬운 에러
2. Logical Error(Runtime Error): 컴파일은 되지만 실행할 때 에러가 발생함. Syntax에러에 비해서 비교적 찾기 어려움. 원하는 연산이 안되어서 원하는 결과가 나오지 않음.

Random Number (난수 생성): 어떤 범위 내에서 generation하는 숫자들의 확률이 동일한 것

컴퓨터의 랜덤 넘버: pseudo 방식. 수학적 랜덤 넘버 생성 방식과 완전히 같지는 않음. seed → 컴퓨터 시간을 입력하면 랜덤 테이블에서 seed 넘버를 찾아서 난수 생성함. ⇒ 실행 시간 기준으로 매번 다른 결과. seed 넘버가 똑같으면 동일한 sequence의 숫자가 나옴(엄밀하게는 정의상 random number는 맞음)

Monte Carlo Simulation

* 장점: 주어진 식 자체가 해석이 안되는 미적분 들어가도 해결할 방법이 존재. 무조건 답을 구할 수 있음.
* 단점: 오차를 줄일 방법이 없음

\*APIs Checklist

- ArrayList

- Concat

- toString

- equals

- Math

- Random

- toUppercase

- toLowerCase

- indexof

- length

-compareTo

-hashcode

-WrapperClass

-getList

-Integer.parseInt

-Math.round

+ CFC와 Collection Interface

Collection Framework Class: 자바에서 제공하는 여러 데이터 구조 클래스

* Tree
* Tree set
* Map
* Array list
* Linked list

CFC는 Collection Interface를 상속하고 있음. 즉, CFC는 Collection Interface의 모든 element들을 realize함. 즉즉, CFC의 모든 클래스들은 모두 동일한 메소드를 사용하고 있다.