2020-11-02

객체란 무엇인가?

의사나 행위가 미치는 대상

구체적, 추상적 데이터의 단위

객체 지향 프로그래밍?

객체를 기반으로 하는 프로그래밍

객체를 정의 하고, 객체의 기능을 구현하며, 객체 간의 협력을 구현

클래스란?

객체를 코드를 구현한 것

객체 지향 프로그래밍의 가장 기본 요소

-멤버 변수

객체가 가지는 속성을 변수로 표현

-메서드

객체의 기능을 구현

Java메모리 구조

Method Area(Static Area)

모든 스레드가 공유하는 영역

JVm 구동 중 사용될 클래스 파일을 읽고 클래스 별로 runtime constant pool(런타임 상수풀)

file data(필드 데이터) , method data(메서드 데이터) ,constructor(생성자) 등을 저장

Static변수

모든 객체가 공유 할 수 있고, 객체 생성 없이 접근 가능

Heap

객체 배열이 생성되는 영역

해당 역역에 생성된 객체와 배열은 JVM 스택 영역의 변수나 다른 객체의 필드에서 참조한다.

참조하는 변수, 필드가 없으면 JVM이 Garbage Collector를 실행하여 해당 객체를 제거한다.

New 연산자로 생성된 객체가 저장되는 영역

JVM 스택영역

Method가 호출 될 때마다 스택 프레임이 생성, 이것이 쌓여 스택을 구성

Method 정보, 로컬변수, 매개변수, 연산중 발생하는 임시데이터 저장

다형성

하나의 코드가 여러 자료형을 구현되어 실행되는 것

같은 코드에서 여러 실행 결과가 나옴

캐스팅

형변환이라고 한다.

업캐스팅

자식 클래스가 부모 클래스의 타입으로 캐스팅되는 것

객체 내에 있는 모든 멤버에 접근할 수 없다. 오직 슈퍼 클래스의 멤버에만 접근 가능하다.

다운캐스팅

부모 클래스가 자식 클래스의 타입으로 캐스팅되는 것

Instanceof를 사용해서 객체의 타입을 구분 할 수있다.

If(변수 instanceof 타입)

주의) instanceof연산자는 참조형 타입에만 사용할 수있다.

String

Concat 와 + 연산자를 사용하게되면 새로운 메모리 주소에 만드는 것이다. 그리고 그것을 가르키는 것이다.

StringBuilder 와 StringBuffer

StringBuffer는 멀티 쓰레드프로그래밍에서 동기화가 보장되고 단일 쓰레드 프로그래밍에서는

StringBuilder를 사용하는 것이 더좋음

2020-11-08

제네릭 프로그래밍

변수의 선언이나 메서드의 매개변수를 하나의 참조 자료형이 아닌 여러 자료형을 변환 될 수 있도록 프로그래밍 하는 방식

실제 사용되는 참조 자료형으로의 변환은 컴파일러가 검증하므로 안정적인 프로그래밍 방식

<T,V> 을 사용 가능 <T extend ?> 사용

컬렉션 프레임워크

프로그램 구현에 필요한 자료구조와 알고리즘을 구현해 놓은 라이브러리

Java.util패키지에 구현

Collection인터페이스와 Map인터페이스로 구성됨

Collection 인터페이스

List 객체를 순서에 따라 저장하고 관리하는데 필요한 메서드가 선언된 인터페이스

ArrayList 배열을 구현한 클래스로 논리적 순서와 물리적 순서가 동일하다.

Vector 두 개의 쓰레드가 동시에 하나의 리소스에 접근 할 때 순서를 맞춰서 동기화 시킴.

LinkedList 논리적으로는 순차적이지만 물리적으로는 순차적이지 않을 수있음 . 추가와 삭제가 용이하다.

Stack LIFO

Queue FIFO

Set :순서대로 저장되지 않기 때문에 iterator()를 사용한다. 중복 X , 순서 X , 저장순서와 출력순서는 다를 수 있따.

HashSet 중복을 허용하지 않으므로 객체의 동일함 여부를 알기위해서 equlas()와 hashCode() 메서드를 재정의해야함

TreeSet 객체의 정렬에 사용되는 클래스

중복을 허용하지 않으면서 오름차순이나 내림차순으로 객체를 정렬함

내부적으로 이진 검색 트리로 구현되어있음

` 객체비교를 위해 Comparable 이나 Comparator인터페이스를 구현해야함

Comparable은 compareTo()메서드를 구현 매개변수와 객체자신을 비교

Comparator는 compare()메서드를 구현 두개의 매개변수를 비교

Map인터페이스

Map key-value pair의 객체를 관리하는데 필요한 메서드가 정의됨, key는 중복 X 내부적으로 hash 방식으로 구현됨(index = hash(key))

Key가 되는 개체는 객체의 유일성함의 여부를 알기위해 equlas()와 hashcode()메서드를 재정의함

Hashtable 동기화를 제공함

Hashmap

TreeMap 객체비교를 위해 Comparable 이나 Comparator인터페이스를 구현해야함

Comparable은 compareTo()메서드를 구현 매개변수와 객체자신을 비교

Comparator는 compare()메서드를 구현 두개의 매개변수를 비교

2020-11-09

내부클래스

클래스 내부에서 사용하기 위해 선언하고 구현하는 클래스

내부에서만 사용하기위해서 사용된다.

인스턴스 내부 클래스

외부 클래스를 먼저 만든 후 내부 클래스 생성

정적 내부 클래스

외부 클래스와 무관하게 생성

지역 내부 클래스

메서드를 호출할 때 생성

익명 내부 클래스

메서드를 호출할 때 생성되거나, 인터페이스 타입변수에 대입할 때 new 예약어를 사용하여 생성

람다식

자바에서 함수형 프로그래밍을 구현하는 방식

클래스를 생성하지 않고 함수의 호출만으로 기능을 수행

함수형 인터페이스를 선언함

내부적으로 익명 객체가 생성되는것이다 .

순수함수를 구현하고 호출

매개 변수만을 사용하도록 만든 함수로 외부 자료에 부수적인 영향이 발생하지 않도록 함

입력 받은 자료를 기반으로 수행되고 외부에 영향을 미치지 않으므로 병렬처리 등에 가능 안정적인 확장성있는 프로그래밍 방식

@FunctionalInterface를 사용하면 함수를 하나만 사용가능하다 .

함수를 변수처럼 사용하는 람다식이다.

2020-11-10

스트림

자료의 대상과 관계없이 동일한 연산을 수행할 수 있는 기능 (자료의 추상화)

한번 생성하고 사용한 스트림은 재사용할 수 없음

스트림 연산은 기존 자료를 변경하지 않음

중간연산과 최종 연산으로 구분된다.

최종 연산이 수행되어야 모든 연산이 적용되는 지연 연산

중간 연산 – filter(),map()

최종연산 – forEach() , count() , sum()

Reduce()연산

정의된 연산이 아닌 프로그래머가 직접 지정하는 연산을 적용

최종 연산으로 스트림의 요소를 소모하며 연산 수행

2020-11-11

예외와 예외처리

컴파일 오류 :프로그램코드 작성 중 발생하는 문법적 오류

실행 오류 :실행 중인 프로그램이 의도 하지 않은 동작을 하거나 프로그램이 중지되는 오류 Runtime error

* 예외처리를 통하여 프로그램의 비정상 종료를 막고 Log를 남길 수있음

시스템 오류 (error) : 가상 머신에서 발생 ,프로그래머가 처리 할 수없음. 예) 동적 메모리를 다 사용한 경우, stack overflow등

예외 :프로그램에서 제어 할 수 있는 오류 예) 읽으려는 파일이 없는 경우 ,소켓오류 ,

모든 예외 클래스의 최상위 클래스는 Exception 클래스 이다 .

try-with-resources문

리소스를 자동으로 해제 하도록 제공해주는 구문 리소스가 AutoCloseable을 구현한 경우 close()를 명시적으로 호출하지않아도 자동으로 호출됨

예외 처리 미루기

Throws 를 사용하여 예외 처리 미루기

try{} 블록으로 예외를 처리 하지 않고, 메서드 선언부에 throws 를 추가

예외가 발생한 메서드에서 예외 처리를 하지않고 이메서드를 호출한 곳에서 예외처리를 한다는 의미

Main()에서 throws를 사용하면 가상 머신에서 처리됨

catch()문에서 Exception 을 먼저걸게되면 아래의 예외가 처리 되지 않는다.

2020-11-20

자바 입출력 스트림

자료의 흐름이 물과 같다는 의미에서 유래

다양한 입출력 장치에 독립적으로 일관성 있는 입출력 방식 제공

I/O대상 기준 : 입력 스트림, 출력 스트림

자료의 종류 : 바이트 스트림, 문자 스트림

스트림의 기능 : 기반 스트림, 보조 스트림

기반 스트림 : 대상에 직접 자료를 읽고 쓰는 기능의 스트림

보조 스트림 : 직접 읽고 쓰는 기능은 없고 추가적인 기능을 제공해 주는 스트림 기반 스트림이나 또 다른 보조

스트림을 생성자의 매개 변수로 표현함

데코레이터 패턴을 이요한다. 기반 스트림 + 보조스트림A + 보조 스트림 B

표준 입출력

System 클래스의 표준 입출력 멤버

PrintStream out , InputStream in , PrintStream err ,

Scanner 클래스

Console클래스

바이트 단위 스트림

InputStream:바이트 단위 입력 스트림 최상위 클래스

OutputStream:바이트 단위 출력 스트림 최상위 클래스

2020-11-23

문자 단위 스트림

Reader

FileReader : 파일에서 문자 단위로 읽는 스트림 클래스

InputStreamReader : 바이트 단위로 읽은 자료를 문자로 변환해 주는 보조 스트림 클래스

BufferedReader : 문자로 읽을 때 배열을 제공하여 한꺼번에 읽을 수 있는 기능을 제공해주는 보조 스트림

Writer

FileWriter : 파일에 문자 단위로 출력하는 스트림 클래스

OutputStreamWriter : 파일에 바이트 단위로 출력한 자료를 문자로 변환해 주는 보조 스트림

BufferedWriter : 문자로 쓸떄 배열을 제공하여 한꺼번에 쓸 수 있는 기능을 제공해 주는 보조 스트림

보조스트림

실제 읽고 쓰는 스트림이 아닌 보조적인 기능을 추가하는 스트림

데코레이터 패턴

직렬화

인스턴스의 상태를 그대로 저장하거나 네트워크로 전송하고 이를 다시 복원하는 방식

ObjectInputStream 과 ObjectOutputStream 사용

보조스트림

Serializable 인터페이스

직렬화는 인스턴스의 내용이 외부로 유출되는 것이므로 프로그래머가 객체의 직렬화 가능 여부를 명시함

Transient -> 이 변수는 직렬화 하지 않을 때 사용

Externalizable 인터페이스

writeExternal , readExternal 을 구현해서 읽고 쓰는 것을 구현해야한다.

그 외 입출력 클래스

File 클래스

RandomAccessFile클래스

입출력 클래스 중 유일하게 파일 입출력을 동시에 할 수 있는 클래스

파일 포인터가 있어서 읽고 쓰는 위치의 이동이 가능함

다양한 자료형에 대한 메서드가 제공됨

2020-11-25

Thread

Process – 실행중인 프로그램 , OS 로부터 메모리를 할당 받음

Thread – 실제 프로그램이 수행되는 작업의 최소단위 , 하나의 프로세스는 하나 이상의 Thread를 가지게 됨

Extend – Thread

Implement- Runnable

Multi-thread 프로그래밍

동시에 여러 개의 Thread가 수행되는 프로그래밍

Thread는 각각의 작업공간(context)를 가진다.

공유 자원이 있는경우 race condition 이 발생

Critical section에 대한 동기화 (Synchronization)의 구현이 필요

동기화구현

Synchronized 수행문과 synchronized 메서드를 이용

Deadlock

Thread status

Start-> Runnable -> (CPU 배분) -> Run

-> (thread 종료 -> Dead

-> sleep(시간),wait(),join() -> Not Runnable - > sleep : 시간이 지나면 , wait() : notify() , join : other thread exits -> Runnalbe

\*wait() -> shared Resource를 기다리는

\*join()->두개의 쓰레드가 동시에 돌아간다고 가정했을 때 한 쓰레드가 다른 쓰레드에게 Join을 걸면 다른 쓰레드가 끝날 때 까지 조인을 건 쓰레드가 Non Runnable상태

로된다.

Interrupt() – 다른 Thread에 예외를 발생시키는 interrupt를 보냄 , thread 가 join(),sleep(),wait() 메서드에 의해 블록킹 되었다면 다시 runnable상태가 될수 있음

Thread 우선순위

Thread.Min\_PRIORITY(=1) ~ Thread.Max PRIORITY(=10)

디폴트는 Thread.NORM\_PRIORITY ( =5) 를 갖게된다 .

우선 순위가 높은 thread는 CPU를 배분 받을 확률이 높다

Thread 종료하기

Thread.stop()은 사용하지 않음 while문이 있을시 flag를 사용한다.