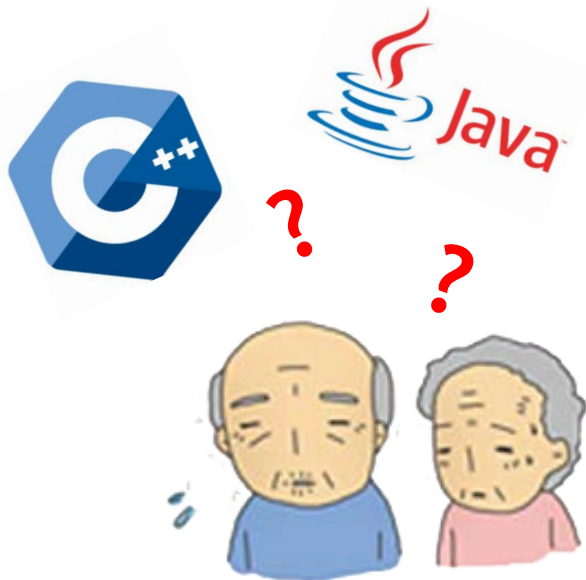


Project#2 Survey on Programing Paradigms

Object Oriented Program (C++ VS JAVA)



제출일: 2016.5.23

과목명: 프로그래밍 언어

소속: 송실대학교 컴퓨터학부

학번: 20122329

발표자: 고완욱

목차

- ▶ C++과 JAVA의 등장
- ▶ C++과 JAVA의 차이점
- ▶ JAVA의 번역과 실행
- ▶ C++과 JAVA의 제공 차이
- ▶ C++ / JAVA 비교정리
- ▶ C++ / JAVA 성능 비교
- ▶ JAVA가 C++보다 느린 원인
- ▶ C++ / JAVA 사용 분야
- ▶ 결론

C++의 등장

▶ 소프트웨어의 위기

- ✓ 하드웨어의 발전에 따라 user들이 요구하는 소프트웨어의 기준이 높아짐
- ✓ 기존 절차 지향 프로그래밍 방식은 규모가 커질수록 유지보수에 한계

▶ 객체 지향 언어의 주목

- ✓ 절차 지향 프로그래밍 방식의 결점을 보완하기 위한 방법으로 객체 지향 프로그래밍 방식에 주목
- ✓ 1979년 벨 연구소에서 C와 시뮬라의 클래스를 접목시켜 C with Class를 만듦
- ✓ 1983년 C++로 개명

JAVA의 등장

▶ Oak 언어

- ✓ C++의 장점을 그대로 가지면서 단점을 보완한 새 언어 개발
- ✓ 가전제품이나 소형기기에 사용을 위해 개발

▶ Java

- ✓ 90년대 www(월드 와이드 웹)의 등장으로 가전제품이 아닌 인터넷에 사용하기 위한 언어로 이름을 java로 바꿈
- ✓ 1996년 1월에 정식버전 발표

초기의 C++ vs JAVA

▶ JVM 머신의 이용

- ✓ 1990년대 전화모뎀을 활용한 정보통신을 하는 것은 매우 느리고 버그가 많았음
- ✓ JAVA는 JVM 머신을 이용해 통신속도와 버그 등을 개선

▶ 오픈소스 정책

- ✓ 기존의 C++은 개발자들이 소스 공유의 예로 사항이 많았음, 라이브러리에 손댈 수 없어 없었음
- ✓ JAVA는 오픈소스 정책, 자주 쓰는 모듈의 공유, 기술 공개, 개발 규칙을 정리해 인기를 얻기 시작함

C++과 Java의 차이점(1)

- ▶ *C++은 함수의 집합, java는 클래스의 집합*
 - ✓ 자바 프로그램의 모든 구성 요소들은 클래스로 구성되며, 특정 클래스 내에서 정의되고 실행됨
- ▶ *C++은 함수, JAVA는 메소드*
 - ✓ 데이터들을 조작하기 위한 행위들을 C++에서는 함수, JAVA에서는 메소드라 함
- ▶ *C++에는 포인터, JAVA는 reference*
- ▶ *C++은 다중 상속, JAVA는 비 다중 상속*
 - ✓ JAVA는 다중상속이 안되는 대신 인터페이스 개념을 제공

C++과 Java의 차이점(2)

▶ C++은 수동적 메모리 관리, java는 자동적 메모리 관리

- ✓ C++은 메모리 등과 같은 자원들의 관리가 가능
- ✓ Java에서 블록의 실행이 닫는 (' } ') 에 오면 list1은 메모리의 영역을 벗어난 것이 되므로 garbage collector가 자동으로 실행하여 이 메모리 영역을 프리 폴로 포함시킴

C++은 자동으로 메모리 쓰레기 수집기능이 없어 delete를 써줘야 함

JAVA

```
{  
    ...  
    List list1;  
    list1 = new List();  
    ....  
}
```

C++

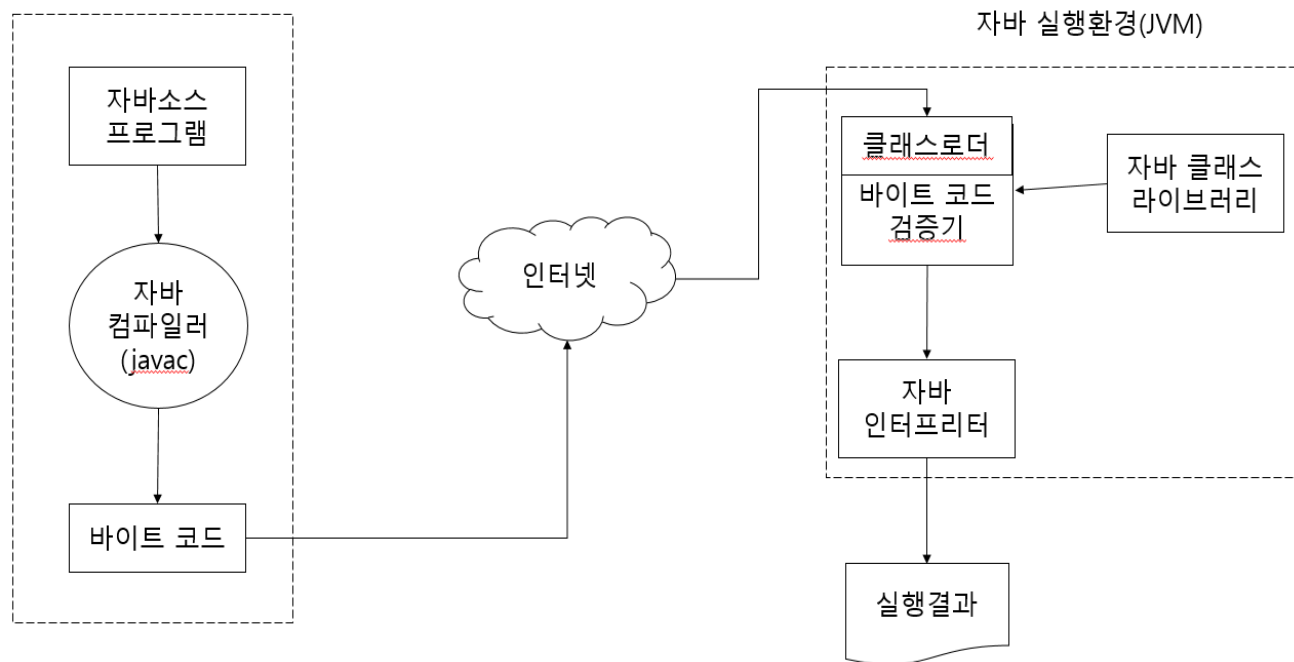
```
{  
    ...  
    List *list1;  
    list1 = new List;  
    ...  
    delete list1;  
}
```

C++과 Java의 차이점(3)

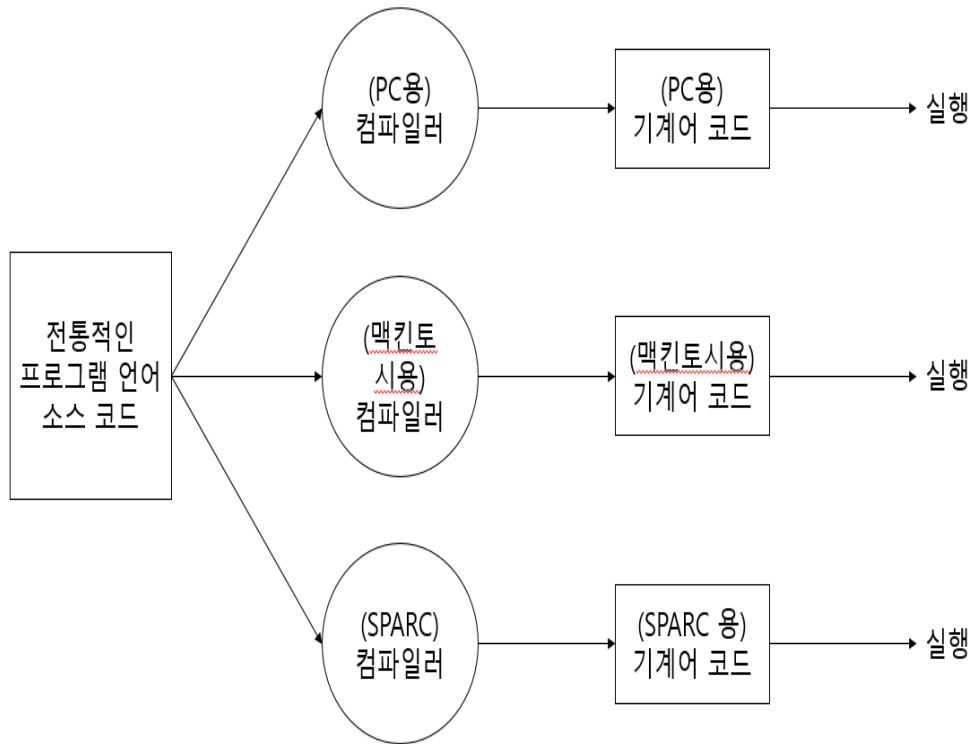
- ▶ *C++은 응용프로그램 JAVA는 응용 및 애플릿 프로그램용*
 - ✓ C++은 독립된 컴퓨터 환경에서 실행되는 응용프로그램을 작성
 - ✓ JAVA는 독립 컴퓨터 환경 뿐만 아니라 인터넷의 다른 컴퓨터 환경에서도 실행될 수 있도록 하는 애플릿 프로그램으로도 작성

JAVA의 번역과 실행(1)

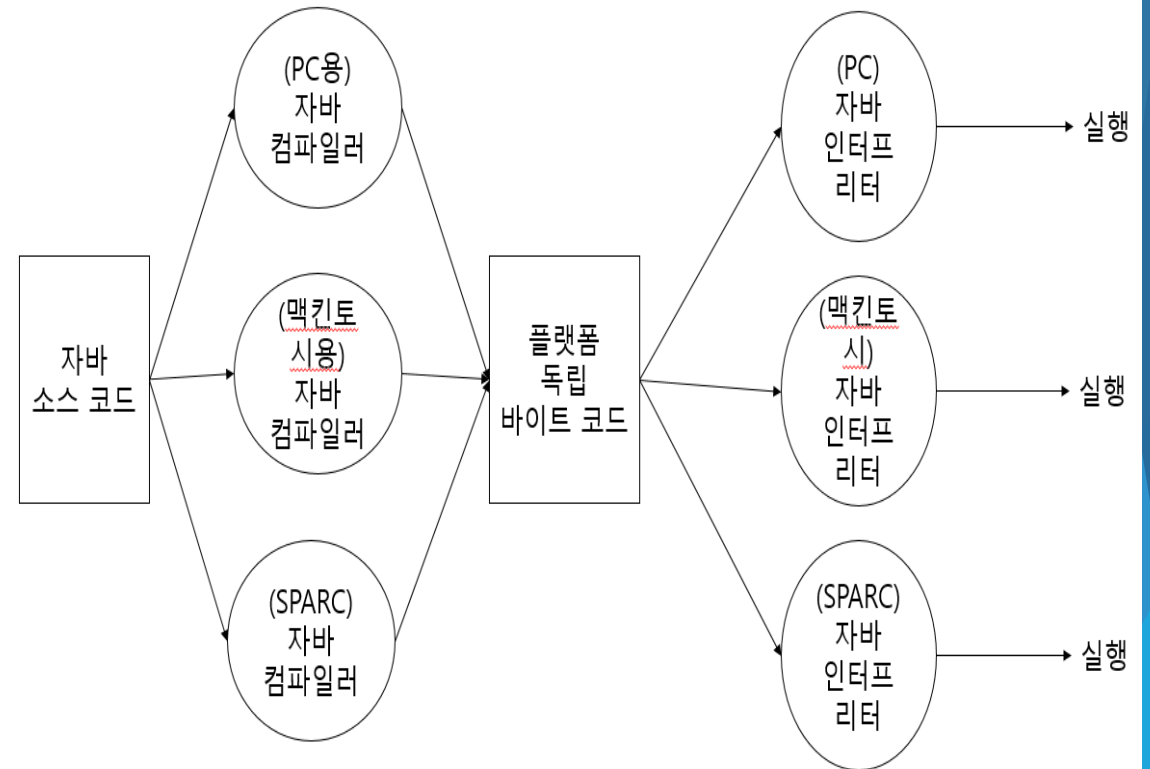
- ▶ C++과 달리 자바는 응용 프로그램 뿐만 아니라 인터넷 환경의 실행을 고려한 언어로
기존과 다른 번역과 실행 방식을 채택
 - ✓ 바이트 코드라는 새로운 코드 방식을 이용
 - ✓ 자바의 실행 환경은 개발 환경과 실행 환경(JVM)을 분리하고 있음
 - ✓ JVM을 장착된 어떤 시스템에서든 실행 가능
 - ✓ JVM은 클래스 로더, 바이트코드 검증기를 포함



JAVA의 번역과 실행(2)



C++(기존)



JAVA

범위지정연산자

▶ 범위지정연산자 (::)

- ✓ 자바에는 없는 것, 클래스의 자료멤버, 멤버함수 또는 전역변수를 참조하기 위해서 사용

```
int A::*pvalue;  
pvalue = &A::value;  
int(A::*getMax)(int, int);  
getMax = &A::max;
```

Pvalue는 클래스 A의 포인터

getMax는 클래스 A의 어떤 함수에 대한 포인터

분기문 (if - else)

▶ 분기문 if

- ✓ C++에서는 조건문의 시험은 숫자 '0'(거짓) 또는 0이 아닌 값(참)으로 결과를 나타냄
- ✓ JAVA에서는 조건문은 true, false 값 중의 하나를 가지는 이진형으로 평가되어야 함

```
int value;  
value=1;  
if(value){  
    System.out.print("Is true");  
}
```

다음은 자바에서 실행하게
되면 error가 발생한다.

접근 수정자

▶ JAVA에서의 접근수정자

- ✓ JAVA는 클래스 이름이나 함수 앞부분에 접근수정자를 둘 수 있음.
- ✓ 각각의 경우에 대해 작성해줘야 함

▶ C++에서의 접근수정자

- ✓ C++은 접근수정자 **public**이나 **private** 뒤에 : 를 함께 표시
- ✓ 그 영역내의 멤버들에게만 영향을 미침

JAVA

```
public class OrderPair{
    public OrderPair(int x_value, int y_value){
        setX(x_value);
        setY(y_value);
    }

    public void setX(int x_value){
        x=x_value;
    }

    public void setY(int y_value){
        y=y_value;
    }

    .....
}
```

C++

```
class OrderPair{
public:
    OrderPair(int x_value, int y_value){
        setX(x_value);
        setY(y_value);
    }
    void setX(int x_value){
        x=x_value;
    }
    void setY(int y_value){
        y=y_value;
    }

    .....
}
```

Package, namespace

▶ JAVA Package

- ✓ 관련 클래스들의 논리적 그룹 또는 집합
- ✓ 특정 문제를 해결하기 위해 프로그래머 자신의 클래스들을 구성하고 프로그래머들이 이미 개발된 클래스들을 쉽게 사용할 수 있도록 함

▶ C++ namespace

- ✓ JAVA의 package와 유사
- ✓ Namespace와 범위지정연산자를 사용하는 것이 귀찮아 이를 위한 using namespace가 있음
- ✓ Namespace는 별칭을 사용하는 데에도 쓰임

```
Pets::Bird::Bird(String nm, int sx, double cost);  
name(nm), gender(sx), sales_price(cost){  
}  
.....  
void Pets::Bird::getData(){  
    cout<<"Pet Data"<<endl;  
    cout<<"Species = "<<name<<endl;  
    .....  
}  
  
using namespace Pets;  
Bird::Bird(String nm, int sx, double cost);  
name(nm), gender(sx), sales_price(cost){  
}  
.....  
void Bird::getData(){  
    cout<<"Pet Data"<<endl;  
    cout<<"Species = "<<name<<endl;  
    .....  
}  
}
```

```
void CreateHawk(){  
    using namespace Zoology;  
    using namespace Pets;  
  
    Bird hawk("Hawk", "Flying predators", 5.0,25);  
    .....  
}
```

단, 사용할 때 애매하게
사용하지 말아야 한다.

추상클래스

- ▶ 추상클래스
 - ✓ 단지 이름의 형식(프로토타입)만을 포함하는 클래스
- ▶ *C++ - virtual*
 - ✓ 순수 가상함수를 포함하는 클래스
- ▶ *JAVA - abstract*
 - ✓ 수정자로도 적용 됨
 - ✓ 추상 클래스로는 어떠한 객체도 생성할 수 없음, 오류 발생

abstract class Shape

Shape some_shape = new Shape() -> 오류

JAVA에서의 생성자(1)

▶ 생성자

- ✓ 객체 생성시 해당 객체의 인스턴스 변수 또는 데이터 멤버들을 초기화할 목적으로만 사용되는 특별한 함수

▶ JAVA와 c++ 모두 생성자를 제공, 작성방법에는 차이

- ✓ Someone이라는 참조 변수만 생성

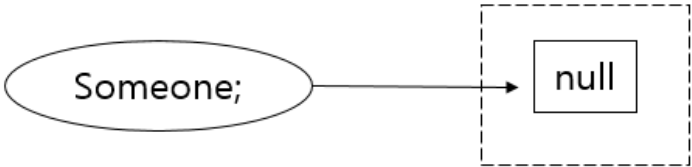
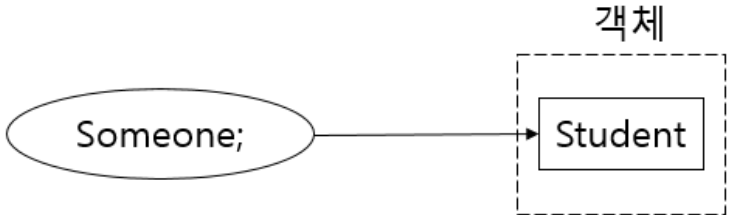
```
Student someone;
```

- ✓ 이 변수가 실제 객체의 물리적 실체를 가지기 위해 생성자와 함께 new 연산자를 사용해야 함

```
Student someone = new Student();
```


JAVA에서의 생성자(2)

- ▶ New 연산자는 객체 생성시 heap 메모리 영역을 사용함

new 실행전 메모리 상태	new 실행 후 메모리 상태
Student someone;	Student someone; someone = new Student();
 <p>A diagram showing an oval labeled 'Someone;' with an arrow pointing to a dashed box containing the word 'null'.</p>	 <p>A diagram showing an oval labeled 'Someone;' with an arrow pointing to a dashed box containing the word 'Student'. Above the dashed box is the label '객체' (Object).</p>

C++에서의 생성자

- ▶ *JAVA에서 사용한 new 연산자는 사용하지 않음*
- ▶ *JAVA와는 달리 생성자 함수들은 초기자 목록을 제공할 수 있음*
- ▶ *초기자 목록은 데이터 멤버들을 한번에 메모리에 할당하고 동시에 초기화할 수 있도록 하기 때문에 성능향상을 도모할 수 있음*

```
class Student{  
    public:  
        Student();    //생성자1  
        Student(String, int, String, char);    //생성자2  
        void Display();  
    private:  
        String name;  
        int ssn;  
        String birthdate;  
        char sex;  
  
};  
  
.....  
Student::Student(String nm, int id, String bd, char sx){ //생성자2  
    :name(String(nm)), ssn(id), birthdate(String(bd)), sex(sx){  
}  
.....  
}
```

소멸자 함수, finalize

▶ C++소멸자 함수

- ✓ 클래스를 사용하고 종료할 때 사용된 자원들을 운영체제에 반환하기 위한 함수
- ✓ 해당 클래스 이름과 ~를 표기

▶ JAVA의 finalize

- ✓ Garbage collector가 자동으로 반환해줌
- ✓ 파일 지정자와 데이터베이스 연결 등과 같은 자원 반환은 수동적으로 해야 해서 finalize라는 메소드를 제공

```
C++      class Publication{
          public:
              Publication(char *, char *);
              ~Publication();    //소멸자 프로토타입
          private:
              char * title;
              char * subject;
      };

      Publication::Publication(char *tit, char *sub){
          ....
      }

      Publication::~~Publication(){
          delete [] title;
          delete [] subject;
      }
```

다중 상속과 인터페이스(1)

▶ C++ 다중 상속

- ✓ 특정 파생 클래스가 여러 개의 상위 클래스들로부터 상속 받는 속성
- ✓ C++에서의 다중상속의 문제점

```
class Transportation{
public:
    ....
    String getName();
private:
    String name;
    ....
}
String Transportation::getName(){
    return name;
}
```

```
cout<<emp.getName()<<"s transportation has accumulated"<<emp.getDistance()<<" miles"<<endl;
```

오류 발생 -> getName()호출이
Employee::getName()인지
Transportation::getName()인지 컴파일
러가 인식을 못함

다중 상속과 인터페이스(2)

▶ JAVA의 인터페이스

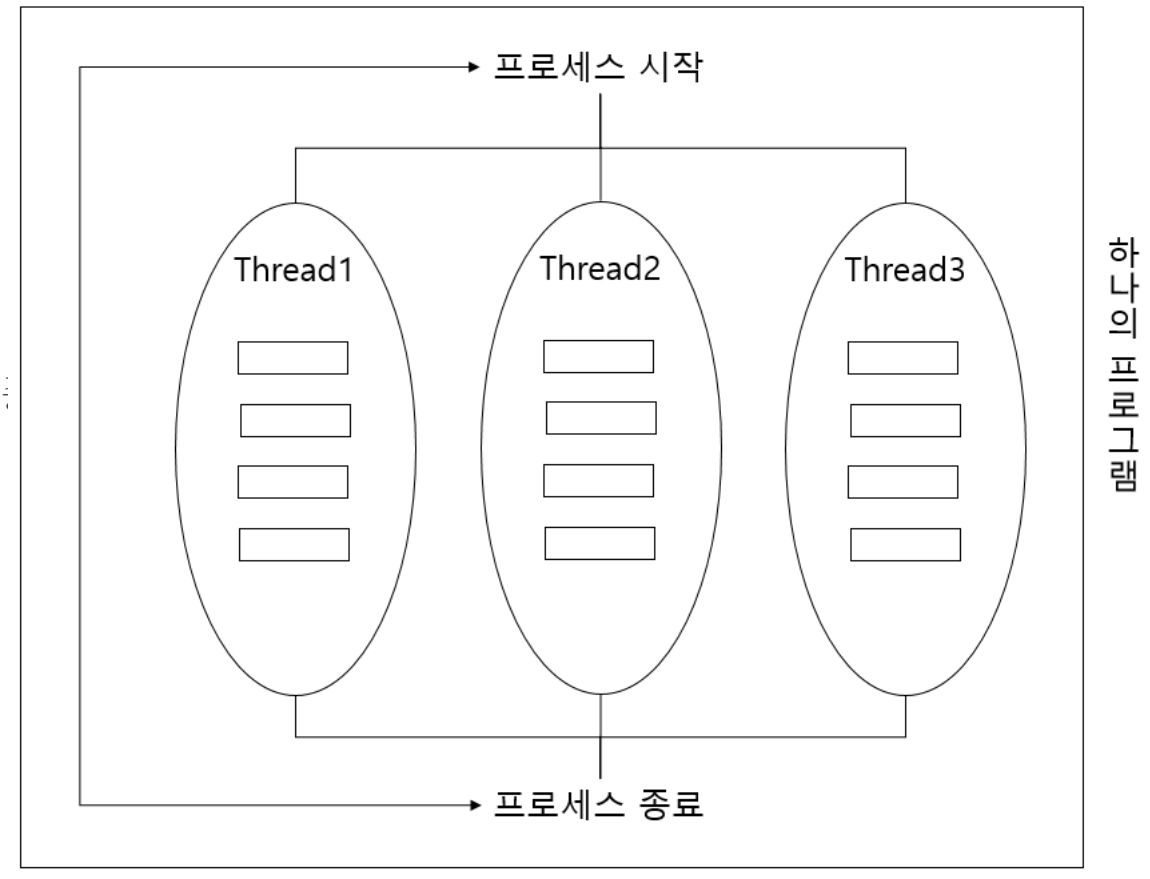
- ✓ 메소드 선언들의 집합
- ✓ 각 메소드 선언들은 메소드의 이름과 반환형만으로 구성
- ✓ 이것의 객체를 생성 불가
- ✓ Implements 키워드를 써서 적용

```
interface Transportation{  
    public long getDistance();  
    public void travel(long dist);  
}
```

```
.....  
class TravelingEmployee extends Employee implements Transportation{  
    TravelingEmployee(String nm, long id, char sx, long dist){  
        super(nm,id,sx);  
        distance=dist;  
    }  
}  
.....
```

Thread(1)

- ▶ JAVA는 c++에 없는 thread를 지원
- ▶ Thread
 - ✓ 프로세스에서 순차적으로 동작하는 문장들의 단일집합
 - ✓ 프로세스는 하나의 Thread를 가져야 함
 - ✓ 하나의 프로세스는 다수의 Thread를 가질 수 있음 -> 다중 Thread
 - ✓ 다중 Thread를 지원하기 위해 표준 라이브러리 패키지(java.lang)의 일부로 thread 클래스를 제공



Thread(2)

▶ JAVA에서의 Thread 생성

- ✓ Thread 클래스에서 직접 상속받아 생성하는 방법

```
public class QSort extends Thread{
    private void Partition(String [] str, int left, int right){
        int index1;
        int index2;
        .....
    }

    .....
    public void run(){
        String line=null;
        int coun=0;
        int index;
        .....
    }
}
```

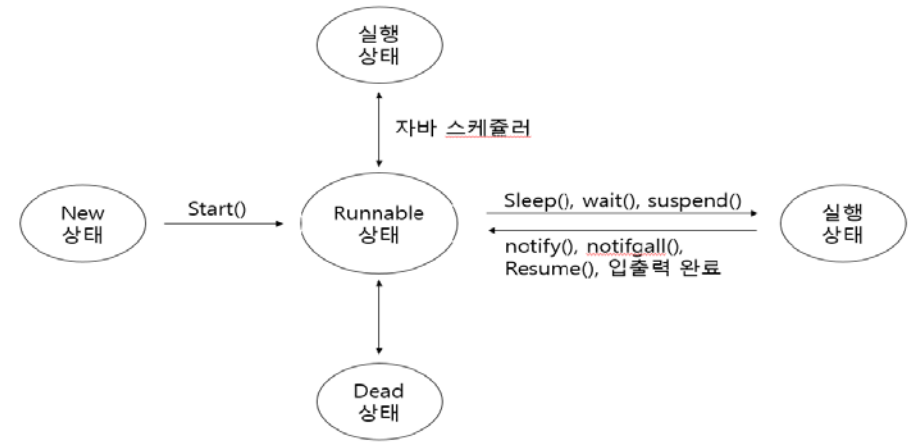
- ✓ Thread 기능을 위한 Runnable 인터페이스를 이용하는 방법

```
public interface Runnable extends Object{
    public abstract void run();
}
```

```
public class QSort implements Runnable{
    private void Partition(String [] str, int left, int right){
        int index1;
        int index2;
        .....
    }

    .....
    System.out.println(Thread.currentThread().getName()+" yiedling...");
    Tread.yield();
    .....
}
```

Thread(3)



▶ Thread의 4가지 상태

- ✓ New 상태는 start() 메시지를 보낸 적이 없는 상태
- ✓ 실행가능 상태는 start()메시지가 thread 객체에 전달되어 객체에 대한 정보가 확립된 상태
- ✓ 대기 상태는 sleep() wait() 등의 메시지가 Thread에 전달되면 대기상태
- ✓ 완료 상태는 Thread의 run()메소드가 정상적으로 실행 완료됐거나 stop() 메시지에 의해 종료 됐을 때

▶ Synchronized

- ✓ 한 Thread가 어떤 메소드를 실행하면, 다른 Thread는 이 메소드를 실행 못하게 끔 함
- ✓ Synchronized 키워드를 가진 메소드로 JAVA는 이 메소드를 실행하고 하는 Thread들을 wait que에 설정해 놓음
- ✓ 실행 Thread가 메소드를 끝내고 나면 que에서 다음 Thread를 선택해 실행
- ✓ 2개 이상의 Thread들이 공유 자원에 대한 수정을 할 때마다 이들 각각의 시도가 충돌아 일어나지 않도록 해줌

C++ / JAVA 비교 정리

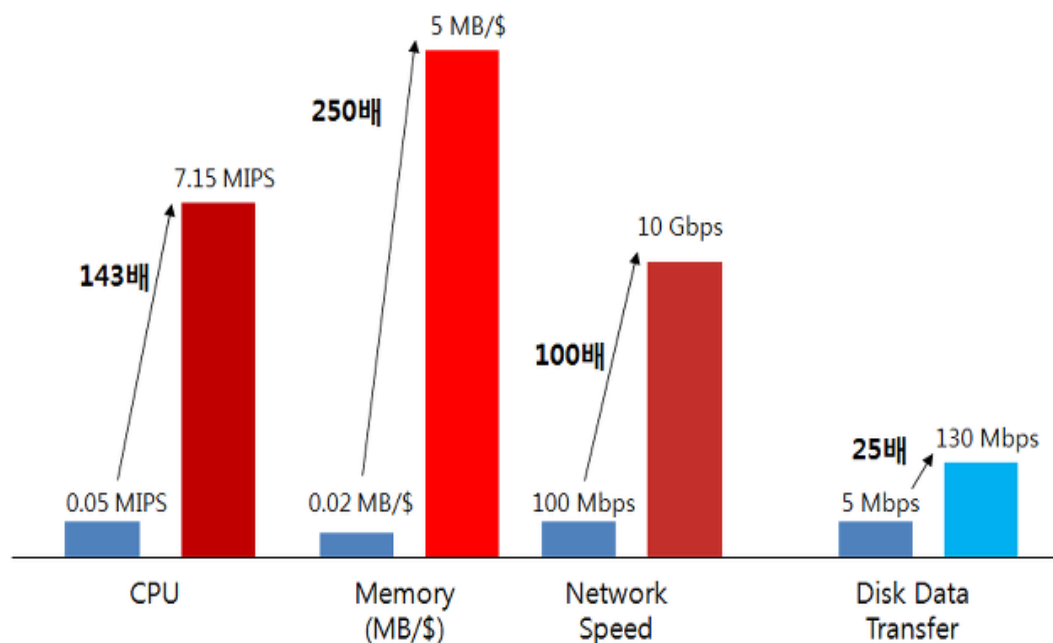
C++

- ▶ 실제 컴퓨터 하드웨어에서 실행 됨
 - ✓ Native 코드를 만들어냄 -> 속도가 빠름
- ▶ 컴파일된 exe 파일을 다른 OS에서 실행시킬 수 없음
- ▶ 웹브라우저에서 C++로 만들어진 프로그램을 실행시킬 수 없음
 - ✓ MS의 Visual c++에서 ActiveX라는 방법을 제공해서 실행시킬수 있으나 보안에 취약하고 윈도우가 아닌 운영체제에서는 작동하지 않은 문제가 있음
- ▶ 포인터 등의 복잡한 문제가 많아 개발 속도가 아주 느리고 버그가 많음

JAVA

- ▶ 가상 머신에서 실행 됨
 - ✓ 가상 머신이라는 한 단계를 더 거쳐 속도가 느림
- ▶ OS의 종류를 가리지 않음
 - ✓ 실제 하드웨어가 아닌 가상 머신 속에서 간접적으로 실행되기 때문
- ▶ 웹브라우저에서 애플릿 형태로 JAVA 프로그램을 실행할 수 있음
- ▶ 포인터 개념이 없어 개발 속도가 빠름

C++ / JAVA 성능 비교



* MIPS (Million Instructions Per Second)



구분(소스코드 링크)	2-WAY PARTITION	3-WAY PARTITION	BUILT-IN
C++ x86	56.0ms	51.6ms	57.8ms
C++ x64	56.3ms	49.9ms	56.3ms
C# x86	90.3ms	69.3ms	86.3ms
C# x64	80.2ms	67.3ms	67.3ms
Java	82.7ms	51.5ms	56.4ms
Node.js	111.0ms	120.2ms	190.5ms

<Quicksort 정렬 속도>

- 1,000,000개의 정수형 난수 데이터를 정렬하는데 소요되는 시간을 측정
- C++의 속도 효율성은 JAVA에 약 1.7~1.9배 속도의 우위에 있음

JAVA가 C++ 보다 느린 원인(1)

- ▶ 모든 오브젝트가 heap에 할당
 - ✓ 자바는 기본 자료형만 stack에 할당하고 모든 오브젝트는 heap에 할당
 - ✓ C++도 큰 오브젝트들을 heap에 할당하지만, JAVA는 작은 오브젝트들도 다 heap에 할당
 - ✓ 그러므로 비용 0의 stack 할당보다 상수 시간이 걸리는 heap에 많이 할당하므로 비용이 많이 듦
- ▶ 엄청 많은 형 변환 cast
 - ✓ JAVA 코드는 형변환 cast로 넘쳐남
 - ✓ JAVA의 형변환은 동적 dynamic 형변환이고, 이것은 많은 비용을 발생 시킴

```
javaDestinationClass makeCast (Object o, Class destinationClass)
{
    Class sourceClass = o.getClass(); // JIT compile-time
    int sourceClassId = sourceClass.getId(); // JIT compile-time

    int destinationId = destinationClass.getId();

    int offset = ourTable [sourceClassId] [destinationClassId];
    if (offset != ILLEGAL_OFFSET_VALUE)
    {
        return ;
    } else throw new IllegalCastException();
}
```

작은 형 변환을 위해 너무 많은
코드를 씬

JAVA가 C++ 보다 느린 원인(2)

▶ 메모리 사용량의 증가

- ✓ 앞서 말한 것처럼 오브젝트들을 heap에 할당하기 때문에 메모리 사용량이 많음
- ✓ JAVA의 오브젝트는 C++에 비해 더 큰데, 이 모든 오브젝트가 virtual table을 가지는 데다가 synchronization primitive를 지원하기 때문에 메모리 사용량이 많음

▶ 디테일을 다루는 컨트롤의 부재

- ✓ JAVA 는 심플한 언어로 설계됨
- ✓ C++은 locality of reference를 향상시킬 수 있는 기능을 제공한다. 그 중 하나로 많은 오브젝트들을 한 번에 할당하고 해제하는 기능도 제공함
- ✓ 이와 같이 c++은 속도를 향상시킬 수 있는 기능 제공 하는 데에 반에 JAVA는 제공하지 않음

C++ / JAVA 사용 분야

C++

- ▶ C++ 컴파일러는 대부분 사용 프로그램에 유리
- ▶ 판매용 프로그램(일반적 게임)
- ▶ 속도를 중요시하는 프로그램 개발 시
 - ✓ 대규모 MMO (Massively Multiplayer Online) 서버 개발
 - ✓ 실시간 서비스 프로그램
 - ✓ 영상처리 관련 프로그램
 - ✓ 암호해독 등 최대한의 연산 속도가 필요한 계산 프로그램에 적합

JAVA

- ▶ 보안, 안전, 호환성을 가장 중요시하는 언어
 - ✓ 기밀이나 중요사항을 많이 다루는 회사나 연구소 내부에서 사용
- ▶ 개발속도를 중요시 하는 프로그램
 - ✓ 응용프로그램 쪽
- ▶ 문법적 통일, 가상 머신 에서의 안정적인 실행
 - ✓ 수학, 과학, 공학적 계산 프로그램에 적합

결론

- ▶ C++과 JAVA는 서로 없는 기능들이 있다.
- ▶ C++이 JAVA보다 성능이 좋다, 즉 실행속도가 빠르다
- ▶ 개인적인 생각
 - ✓ 어떤 때에는 C++를 쓰지 JAVA를 쓰지는 그 project가 실시간이나 프로그램 속도를 중요시 하는 것인지, 보안이나 호환성 및 빠르게 개발을 할 때인지를 파악하여 C++을 쓰지 JAVA를 쓰지를 결정 해야 한다.

즉, 결국 프로그래머의 언어 선택의 안목이 중요하다....

Thank You!



-참조 문헌: (객체지향 개념으로 본)
자바와 c++ 모두 잡기