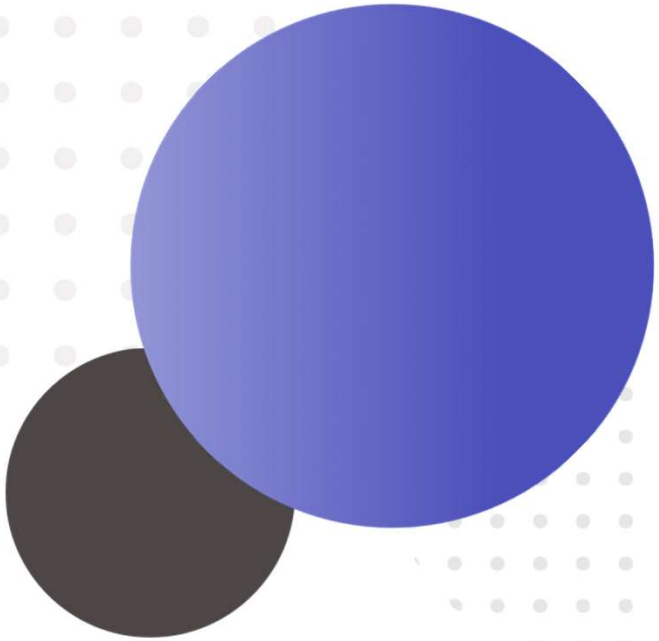


능동적 사고 방식의

java

강사 박주병

Park Ju Byeong

Part14 자료구조와 쓰레드

01 열거형

02 자료구조

03 쓰레드

04 실습 문제

01

열거형

1-1 실습문제 (normal)

People 클래스를 만들고 열거형을 사용하여 취미를 저장하는 멤버변수를 만들어 보자.
- 취미는 soccer,baseball,cook,running 가 있다.

```
People p1 = new People();

p1.hobby = People.HOBBY.BASEBALL;
System.out.println(p1.hobby);
```

```
<terminated> Main (4
RUNNING
```


Park Ju Byeong

1-1 문제풀이 (normal)

```
public class People {

    public enum HOBBY{SOCCER,BASEBALL,COOK,RUNNING;}

    public HOBBY hobby;

}
```


Park Ju Byeong

02

자료구조

2-1 실습문제 (normal)

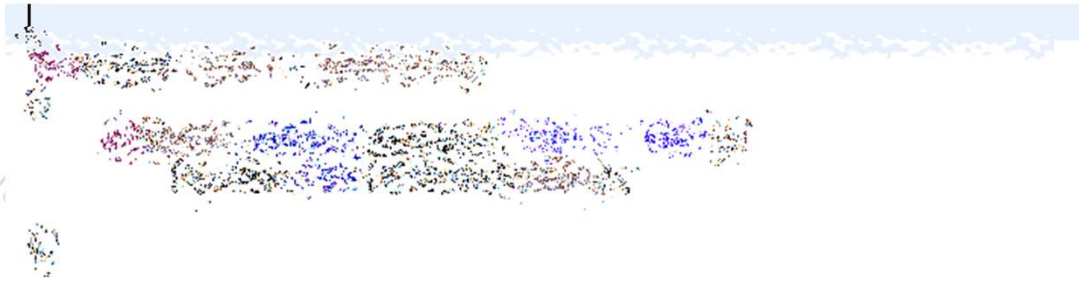
ArrayList를 사용하여 도서의 제목과 저자를 관리하는 프로그램을 만들어 보자. 각 도서는 Book 클래스 의 객체로 관리된다.

ArrayList에 제목과 저자 데이터가 있는 Book 객체 여러 개를 넣어두고 특정 저자의 책들만 출력해보자.

Book 클래스(코드 예시를 보고 이외에 필요한 메서드나 생성자를 적절하게 만들어보자)
- 멤버변수 : String title , String author

```
List<Book> bookList = new ArrayList();

bookList.add(new Book("해리포터와 비밀의방","조앤 k. 롤링"));
bookList.add(new Book("해리포터와 아즈카반의죄수","조앤 k. 롤링"));
bookList.add(new Book("백종원이 추천하는 집밥 메뉴","백종원"));
bookList.add(new Book("몰입","황농문"));
```



Problems Javadoc Search Console Git Staging Hist
<terminated> Main (1) [Java Application] C:₩Users₩zest1₩.p2₩pool₩plugins₩or
제목:해리포터와 비밀의방 저자:조앤 k. 롤링
제목:해리포터와 아즈카반의죄수 저자:조앤 k. 롤링

2-1 문제풀이 (normal)

```
List<Book> bookList = new ArrayList();

bookList.add(new Book("해리포터와 비밀의방","조앤 k. 롤링"));
bookList.add(new Book("해리포터와 아즈카반의죄수","조앤 k. 롤링"));
bookList.add(new Book("백종원이 추천하는 집밥 메뉴","백종원"));
bookList.add(new Book("몰입","황농문"));

for(Book book : bookList)
{
    if(book.author.equals("조앤 k. 롤링"))
        System.out.println(book);

}
```

```
public class Book {


    String title;
    String author;


    Book(String title, String author)
    {
        this.title = title;
        this.author = author;
    }


    @Override
    public String toString() {
        // TODO Auto-generated method stub
        return "제목:"+title+" 저자:"+author;
    }
}
```

2-2 실습문제 (normal)

WordCount 클래스를 만든후 HashMap을 이용하여 단어별로 몇 개가 있는지 카운트하는 기능을 만들어보자.

```
public static void main(String[] args)    {
    // TODO Auto-generated method stub
    WordCount wc = new WordCount();

    wc.add("강아지");
    wc.add("복숭아");
    wc.add("밀키트");
    wc.add("자전거");
    wc.add("운동화");
    wc.add("강아지");
    wc.add("강아지");
    wc.add("운동화");

    System.out.println(wc.getWordCount("강아지"));

    wc.printAllWordCount();

}
```

```
<terminated> Main (6) [Java Application]
3
단어:복숭아    갯수:1
단어:자전거    갯수:1
단어:강아지    갯수:3
단어:밀키트    갯수:1
단어:운동화    갯수:2
```


Park Ju Byeong

2-2 문제풀이 (normal)

```
public class WordCount {

    private HashMap<String,Integer> wordList = new HashMap();


    void add(String word)
    {
        //int cnt = wordList.get(word) == null ? 0 : wordList.get(word) ;

        int cnt = wordList.getOrDefault(word,0);

        wordList.put(word, ++cnt);//기존 단어갯수에 1증가 시켜서 넣는다.
    }

    int getWordCount(String word)
    {
        return wordList.getOrDefault(word,0);
    }

    void printAllWordCount()
    {
        //모든 요소를 순회하며 꺼내올때는 향상된 for문을 이용하고 Set형태로 변환하여 가져온다.
         for (Map.Entry<String, Integer> entry : wordList.entrySet())
         {
            System.out.println("단어:"+entry.getKey() +"\t 갯수:"+ entry.getValue());
         }
    }


}
```

2-3 실습문제 (hard)

제네릭을 활용하여 나만의 MyStack 자료구조를 만들자.

- void push(T value), T pop() 메서드를 구현하자.
- 내부에서 실질적으로 데이터는 ArrayList로 관리하자.

ex) push의 내부는 ArrayList 객체의 add 메서드를 사용하여 요소를 추가한다.

```
MyStack<String> stack = new MyStack<>();

stack.push("1번");
stack.push("2번");
stack.push("3번");

System.out.println(stack.pop());
System.out.println(stack.pop());
System.out.println(stack.pop());
```

```
<terminated>
3번
2번
1번
```

2-3 문제풀이 (hard)

```java
public class MyStack<T> {
    private List<T> stack;

    public MyStack(){
        this.stack = new ArrayList<>();
    }

    public void push(T value){
        stack.add(value);
    }

    public T pop(){
        if(stack.isEmpty())
        {
            throw new RuntimeException("Stack is empty");
        }

        T value = stack.get(stack.size()-1);
        stack.remove(stack.size()-1);
        return value;
    }
}
```


Park Ju Byeong

04

실습문제

3-1 실습문제 (normal)

0~10초까지 카운트를 출력해보자

- 1초에 한번 숫자가 출력되어야 한다.
- 출력하기전 현재의 쓰레드를 1초간 정지 해야 한다.
- Thread.sleep(1000); 을 호출하면 해당 쓰레드는 1초간 멈춘다.

Park Ju Byeong

3-1 문제풀이 (normal)

```
for(int i=0;i<11;i++)
{
    try
    {
        Thread.sleep(1000);
        System.out.println(i);

    }catch(Exception ex)
    {

    }
}
```


Park Ju Byeong

3-2 실습문제 (normal)

카운트가 올라가면서 동시에 다이얼로그 창이 띄워지도록 하자
- 다이얼로그가 띄워진 채 숫자가 올라가야 한다.
- 카운트 출력이 서브쓰레드로 실행이 되어야 가능하다.
- Thread 클래스, Runnable 인터페이스 둘 중 편한 것을 사용하자.

3-2 문제풀이 (normal)

```
new Thread(new Runnable(){
      @Override
     public void run() {

          for(int i=0;i<10;i++)
          {
               try
               {
                    Thread.sleep(1000);
                    System.out.println(i);

               }catch(Exception ex)
               {

               }

          }
     }
  }
  ).start();

JOptionPane.showInputDialog("값을 입력하세요");
```


Park Ju Byeong

3-3 실습문제 (normal)

1번 문제의 카운트가 다이얼 로그창을 닫을 때 까지 계속 나오게 하고
다이얼로그 창을 닫으면 프로그램이 종료되도록 하자

3-3 문제풀이 (normal)

```
new Thread(new Runnable(){

    @Override
    public void run() {

        for(int i=0;i<10;i++)
        {
            try
            {
                Thread.sleep(1000);
                System.out.println(i);
            }catch(Exception ex)
            {

            }

        }
    }
}
).start();


JOptionPane.showInputDialog("값을 입력하세요");
```

```
Thread th = new Thread(new Runnable(){
    @Override
    public void run() {
        int i=0;
        while(true)
        {
            try
            {
                Thread.sleep(1000);
                System.out.println(i);
                i++;
            }catch(Exception ex)
            {

            }

        }
    }
}
);

th.setDaemon(true);
th.start();
JOptionPane.showInputDialog("값을 입력하세요");
```

데몬쓰레드로 설정하여
메인쓰레드가 종료될때 같이
종료된다.

THANK YOU

강사 박주병