14주차 과제 및 실습

과제 및 실습

- 과제는 문제 및 프로그램 실습으로 나뉘어 있습니다.
  - 반드시 프로그램 예제를 실행하고 이해한 후 과제를 수행하세요.

- 제출 파일 : 보고서+소스코드
  - 보고서 : 한글 또는 word 파일
  - 소스코드 : java 파일만 제출
  - 위 두 파일을 학번_이름_실습주차.zip으로 압축하여 제출

- 보고서 내용
  - 과제 문제 : 해답, 해답의 이유
  - 실습 문제 : 해답소스코드, 프로그램 설명, 결과 화면(결과 캡처)
    - 프로그램 설명 : 작성한 소스코드의 내용, 소스코드 내 주석으로 대체 가능
  - 보고서 파일명 : 학번_이름_실습주차.hwp
    - ex) 2023111010_홍길동_2주차.hwp

- 소스코드
  - 파일명 : Exec1_학번_문제번호.java

- 제출 : eclass
  - 과제 연장 제출을 희망할 경우 직접 또는 이메일을 통해 조교에게 요청

프로그램 예제1. 스레드 예제 : Thread 클래스의 동기화(강의 슬라이드의 자료)

- 강의 슬라이드의 예를 작성해보세요.
  - 도서관에 3자리에 학생 5명이 순차적으로 앉으려는 경우(wait(), notify(), notifyAll() 메소드)

```java
package edu.dongguk;

import java.util.ArrayList;

class JavaLibrary {
    public ArrayList<String> arrayList = new ArrayList<String>();

    public JavaLibrary() {
        arrayList.add("1번 자리");
        arrayList.add("2번 자리");
        arrayList.add("3번 자리");
    }

    public synchronized String study() throws InterruptedException {
        Thread t = Thread.currentThread();
        if (arrayList.size() == 0) {
            System.out.println(t.getName() + " : 대기 시작");
            wait();
            System.out.println(t.getName() + " : 대기 끝");
        }

        if (arrayList.size() > 0) {
            String seat = arrayList.remove(0);
            System.out.println(t.getName() + " : " + seat +" 착석함");
            return seat;
        } else return null;
    }

    public synchronized void returnSeat (String seat) {
        Thread t = Thread.currentThread();

        arrayList.add(seat);
        notify();
        System.out.println(t.getName() + " : " + seat + "가 반환됨");
    }
}
```

```java
class Student extends Thread {
    public void run() {
        try {
            String seatNum = Library.library.study();
            if (seatNum == null)
                return;
            sleep (5000);
            Library.library.returnSeat(seatNum);
        }catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}

public class Library {
    public static JavaLibrary library = new JavaLibrary();

    public static void main(String[] args) {
        Student[] std = new Student[5];

        for(int i = 0; i < 5; i++) {
            std[i] = new Student();
            try {
                Thread.sleep(500);
            }catch(InterruptedException e) {}
            std[i].start();
        }
    }
}
```

```
Thread-0 : 1번 자리 착석함
Thread-1 : 2번 자리 착석함
Thread-2 : 3번 자리 착석함
Thread-3 : 대기 시작
Thread-4 : 대기 시작
Thread-0 : 1번 자리가 반환됨
Thread-3 : 대기 끝
Thread-3 : 1번 자리 착석함
Thread-1 : 2번 자리가 반환됨
Thread-4 : 대기 끝
Thread-4 : 2번 자리 착석함
Thread-2 : 3번 자리가 반환됨
Thread-3 : 1번 자리가 반환됨
Thread-4 : 2번 자리가 반환됨
```

프로그램 예제2. Annotation

일반적으로 어노테이션 내의 정보는 프로그램에서 사용되지 않는다.

reflection으로 클래스의 정보를 읽어와 이용한다.

- 어노테이션의 값을 출력

- 사용자 어노테이션 작성

```
import java.lang.annotation.*;

@Target({ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@interface MyAnnotation {
        String value() default "MyAnnotation-기본값";
}

class MyObject {
    @MyAnnotation
    public void testMethod1() {
        System.out.println("메소드1 : 호출");
    }
    @MyAnnotation(value = "새로운 어노테이션값")
    public void testMethod2() {
        System.out.println("메소드2 : 호출");
  }
}

public class AnnotationTest {
    public static void main(String[] args) {
        MyObject m1 = new MyObject();
        m1.testMethod1();
        m1.testMethod2();

        Method[] methodList = MyObject.class.getMethods();//클래스에서 메서드를 읽어옴

        for (Method method : methodList) {
            if(method.isAnnotationPresent(MyAnnotation.class)) {
                MyAnnotation annotation = method.getDeclaredAnnotation(MyAnnotation.class);
                String value = annotation.value();      //메서드에서 어노테이션내 value의 값을 읽어옴
                System.out.println(method.getName() + ":" + value);
            }
        }
    }
}
```

```
메소드1 : 호출
메소드2 : 호출
testMethod1:MyAnnotation-기본값
testMethod2:새로운 어노테이션값
```

과제문제 : 다음의 문제를 풀고 정답의 이유도 함께 작성하세요.

1. 스레드의 등기화의 설명으로 옳지 않은 것은?

① 여러 개의 스레드가 동일한 값에 접근하는 경우 일관성을 위하여 동기화가 필요하다.

② 스레드의 동기화는 접근하는 메서드에만 임계구역을 설정할 수 있다.

③ 임계구역은 synchronized 키워드를 사용하여 만든다.

④ 임계구역내에서 wait()를 통하여 다른 객체에서 사용이 끝나길 기다릴 수 있다.

⑤ wait()에 의해 대기중인 메서드는 notify()메서드로 깨울 수 있다.

2. 자바의 멀티 태스킹에 대한 설명 중 틀린 것은?

① 자바 스레드의 우선순위의 기본값은 값이 5이다.

② 자바에서는 우선 순위가 높은 스레드 일수록 순위값이 값이 크다

③ yield()메서드로 다른 스레드에 실행 순서를 양보할 수 있다.

④ 스레드의 우선순위값은 변경할 수 없다.

과제문제 : 다음의 문제를 풀고 정답의 이유도 함께 작성하세요.

3. 스레드의 설명으로 옳지 않은 것은?

① 자바에서 스레드는 선점형우선순위 알고리즘으로 운영된다.

② JVM의 스케줄러에 의해 실행된다.

③ 스레드가 종료되는 경우 다시 재시작할 수 있다.

④ 스레드는 다른 스레드를 종료할 수 있다.

4. 다른 스레드가 종료할때 까지 기다리는 메소드는?

① run()　② start()　③ notify()　④ wait()　⑤ join()

5.Annotation에 대한 설명으로 옳지 않은 것은?

① Annotation의 값은 프로그램 실행에 영향을 주지 않는다.

② Annotation @으로 시작하며 클래스나 메서드에 정보를 다양한 제공한다.

③ 사용자 지정 Annotation으로 문법적인 오류 발생에 대한 처리를 할 수 있다.

④ 만약 @Override를 지정한 메서드가 있다면 부모에 있는지 여부를 체크를 할 수 있다.

실습 문제1. 스레드 작성하기.

Exec1) 13주차 Exec1를 아래와 같이 고치는 경우 문제점을 해결하고 출력하는 코드를 작성하라.

- 한 계좌에 출금과 입금이 동시에 이루어지는 경우,

계좌 금액의 불일치 오류가 발생되지 않도록 이체와 입금의 경우 코드를 작성하라.

(synchronized로 임계구역 설정 및 사용)

계좌 클래스 Account

- 맴버 금액은 10000원으로 초기화한다.
- 입금, 출금을 위한 메서드를 만들어라.

은행 거래 스레드 클래스 Bank

- 맴버로 위 계좌를 갖는다.
- Bank 스레드 실행 시, 은행거래를 위해 1~1000사이의 Random 값을 생성한다.
- 위 Random 값의 10으로 나눈 나머지 * 1000 ms 만큼 정지(sleep)한다.
- 위 Random 값이 짝수값이면 Random 값 만큼 상대 계좌에 이체를 하고,
  홀수 값이면 Random 값이 입금한다.

Main()

- 은행 거래 클래스 BankEx로 Bank 객체 2개를 생성하고 각 객체의 스레드를 실행시킨다.

※ 다른 계좌에 이체를 하는 메서드 작성.(Account 또는 Bank 클래스 중 적절한 곳에 작성한다)