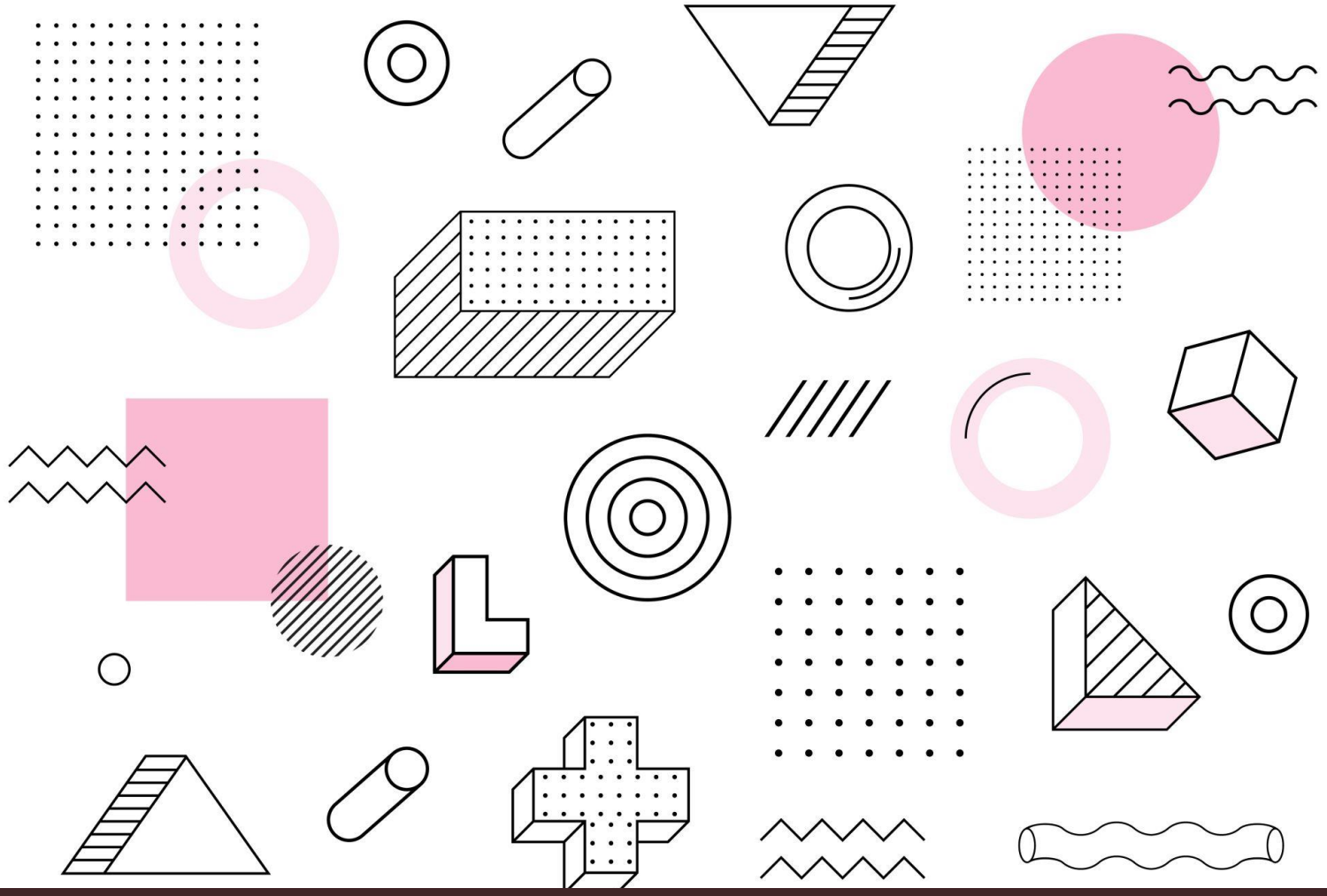

1학기 고급자바 실습

week 8-2(수업 10주차 실습)

김민진(18)

김지희(18)

문의 메일 : genie02166@duksung.ac.kr

Part1. 스레드

1) 스레드 우선순위

2) 스레드 상태 제어

- sleep()
- yield()
- join()

3) notify(), wait() 예제

1) 스레드 우선순위

CalcThread.java | PriorityExample.java | Calculator.ja

```java
package week8_1;

public class CalcThread extends Thread {
    public CalcThread(String name) {
        setName(name);
    }

    public void run() {
        for(int i=0; i<2000000000; i++) {
        }
        System.out.println(getName());
    }
}
```

실행 결과

Problems | Javadoc

<terminated> PriorityExampl

```
thread10
thread1
thread2
thread8
thread4
thread5
thread9
thread6
thread7
thread3
```

PriorityExample.java | Calculator.java | MainThreadExample.java

```java
package week8_1;

public class PriorityExample {
    public static void main(String[] args) {
        for(int i=1; i<=10; i++) {
            Thread thread = new CalcThread("thread" + i);
            if(i != 10) {
                thread.setPriority(Thread.MIN_PRIORITY);
            } else {
                thread.setPriority(Thread.MAX_PRIORITY);
            }
            thread.start();
        }
    }
}
```

2) 스레드 상태 제어

- sleep()

SleepExample.java

```
1  package week8_1;
2
3  import java.awt.Toolkit;
4
5  public class SleepExample {
6      public static void main(String[] args) {
7          Toolkit toolkit = Toolkit.getDefaultToolkit();
8          for(int i=0; i<10; i++) {
9              toolkit.beep();
10             try {
11                 Thread.sleep(3000);
12             } catch(InterruptedException e) {
13             }
14         }
15     }
16 }
```

2) 스레드 상태 제어

- yield()

YieldExample.java | ThreadA.java | ThreadB.java

```java
package week8_1;

public class ThreadA extends Thread {
    public boolean stop = false;
    public boolean work = true;

    public void run() {
        while(!stop) {
            if(work) {
                System.out.println("ThreadA 작업 내용");
            } else {
                Thread.yield();
            }
        }
        System.out.println("ThreadA 종료");
    }
}
```

YieldExample.java | ThreadA.java | ThreadB.java

```java
package week8_1;

public class ThreadB extends Thread {
    public boolean stop = false;
    public boolean work = true;

    public void run() {
        while(!stop) {
            if(work) {
                System.out.println("ThreadB 작업 내용");
            } else {
                Thread.yield();
            }
        }
        System.out.println("ThreadB 종료");
    }
}
```

2) 스레드 상태 제어

- yield()

YieldExample.java | ThreadA.java | ThreadB.java

```java
package week8_1;

public class YieldExample {
    public static void main(String[] args) {
        ThreadA threadA = new ThreadA();
        ThreadB threadB = new ThreadB();
        threadA.start();
        threadB.start();

        try { Thread.sleep(3000); } catch (InterruptedException e) {}
        threadA.work = false;

        try { Thread.sleep(3000); } catch (InterruptedException e) {}
        threadA.work = true;

        try { Thread.sleep(3000); } catch (InterruptedException e) {}
        threadA.stop = true;
        threadB.stop = true;
    }
}
```

실행 결과

Console | Problems

<terminated> YieldExamp

```
ThreadA 작업 내용
ThreadA 작업 내용
ThreadA 작업 내용
ThreadA 작업 내용
ThreadA 작업 내용
ThreadA 작업 내용
ThreadA 작업 내용
ThreadA 작업 내용
ThreadA 작업 내용
ThreadB 작업 내용
ThreadB 작업 내용
ThreadB 작업 내용
ThreadB 작업 내용
ThreadB 작업 내용
ThreadB 작업 내용
ThreadB 작업 내용
ThreadA 작업 내용
ThreadB 종료
ThreadA 종료
```

2) 스레드 상태 제어

- join()

SumThread.java

```java
package week8_1;

public class SumThread extends Thread {
    private long sum;

    public long getSum() {
        return sum;
    }

    public void setSum(long sum) {
        this.sum = sum;
    }

    public void run() {
        for(int i=1; i<=100; i++) {
            sum+=i;
        }
    }
}
```

JoinExample.java

```java
package week8_1;

public class JoinExample {
    public static void main(String[] args) {
        SumThread sumThread = new SumThread();
        sumThread.start();

        try {
            sumThread.join();
        } catch (InterruptedException e) {
        }

        System.out.println("1~100 합: " + sumThread.getSum());
    }
}
```

실행 결과

Console

```
<terminated> JoinEx
1~100 합: 5050
```

3) notify(), wait() 예제

DataBox.java | ProducerThread.java | ConsumerThread.java | WaitNotifyExample.java

```java
package week8_1_3;

public class DataBox {
    private String data;

    public synchronized String getData() {
        if(this.data == null) {
            try {
                wait();
            } catch(InterruptedException e) {}
        }
        String returnValue = data;
        System.out.println("ConsummerThread가 읽은 데이터: " + returnValue);
        data = null;
        notify();
        return returnValue;
    }

    public synchronized void setData(String data) {
        if(this.data != null) {
            try {
                wait();
            } catch(InterruptedException e) {}
        }
        this.data = data;
        System.out.println("ProducerThread가 생성한 데이터: " + data);
        notify();
    }
}
```

3) notify(), wait() 예제

DataBox.java | ProducerThread.java | ConsumerThread.java

```java
package week8_1_3;

public class ProducerThread extends Thread {
    private DataBox dataBox;

    public ProducerThread(DataBox dataBox) {
        this.dataBox = dataBox;
    }

    @Override
    public void run() {
        for(int i=1; i<=3; i++) {
            String data = "Data-" + i;
            dataBox.setData(data);
        }
    }
}
```

DataBox.java | ProducerThread.java | ConsumerThread.java

```java
package week8_1_3;

public class ConsumerThread extends Thread {
    private DataBox dataBox;

    public ConsumerThread(DataBox dataBox) {
        this.dataBox = dataBox;
    }

    @Override
    public void run() {
        for(int i=1; i<=3; i++) {
            String data = dataBox.getData();
        }
    }
}
```

3) notify(), wait() 예제

DataBox.java | ProducerThread.java | ConsumerThread.java | WaitNotifyExample.java

```
1  package week8_1_3;
2
3  public class WaitNotifyExample {
4      public static void main(String[] args) {
5          DataBox dataBox = new DataBox();
6
7          ProducerThread producerThread = new ProducerThread(dataBox);
8          ConsumerThread consumerThread = new ConsumerThread(dataBox);
9
10         producerThread.start();
11         consumerThread.start();
12     }
13 }
```

실행 결과

Console | Problems | Javadoc | Declarati

<terminated> WaitNotifyExample (2) [Java Applic

```
ProducerThread가 생성한 데이터: Data-1
ConsummerThread가 읽은 데이터: Data-1
ProducerThread가 생성한 데이터: Data-2
ConsummerThread가 읽은 데이터: Data-2
ProducerThread가 생성한 데이터: Data-3
ConsummerThread가 읽은 데이터: Data-3
```

출석 과제 (5/21 금 오후 11:55 마감)

Q. 스레드 우선순위에 대한 설명 중 틀린 것?

1) 우선순위가 높은 스레드가 실행 기회를 더 가진다.
2) 우선 순위는 1부터 10까지이다.
3) Thread 클래스는 우선순위 상수를 제공한다.
4) 1은 가장 높은 우선순위라 다른 스레드보다 실행 기회를 더 많이 갖게 한다.

위의 기간까지 고급 자바 실습
DS-CLASS 에 제출하시기 바랍니다.