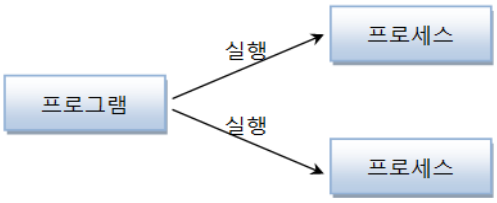
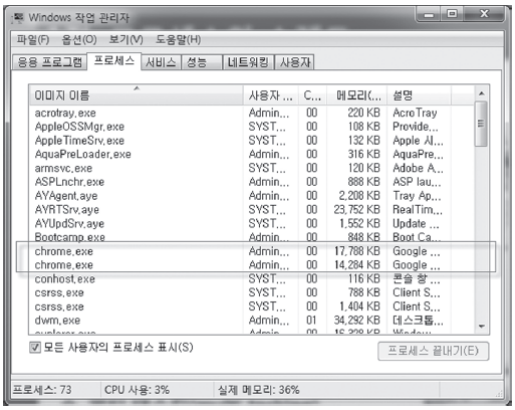
쓰레드 (Thread)





쓰레드는 멀티 태스킹을 지원해 주는 기술이다.

Thread를 생성시키는 방법은 2가지가 있다. Thread를 상속받아 run()을 재정의 하는 방법이 있고, Runnable interface를 implements 하여 Thread 클래스를 이용하여 객체를 만들고 start()를 호출하면 된다. 주의할 점은, Runnable 은 start 메서드를 가지고 있지 않기 때문에 Thread를 한번 더 사용해야 한다.

실습예제1

Thread 클래스를 상속받아 run() 메서드를 구현하기

```
public class ThreadEx1 {
	public static void main(String[] args) {
		Thread t1 = new MyThread("A");
		Thread t2 = new MyThread("B");
		Thread t3 = new MyThread("C");

		t1.start();
		t2.start();
		t3.start();
	}
}

class MyThread extends Thread {
	private String name;

	MyThread(String name) {
		this.name = name;
	}

	@Override
	public void run() {
		// 스레드로 수행될 코드
		for(int i=0; i<10; i++) {
			System.out.print(name + " ");
```

```
                    try {
                        Thread.sleep(10);
                    } catch (InterruptedException e) {
                    }
                }
        }
}
```

아래 실행 결과를 보면 A, B, C가 섞여서 출력된다.

A B C C A B B A C A C B B A C A C B B A C C A B C A B B A C

실습예제2

Runnable 인터페이스를 implements 하고Thread 클래스를 이용하여 run() 메서드 구현하기

```
public class ThreadEx2 {
        public static void main(String[] args) {
                Thread t1 = new Thread(new MyThread2("A"));
                Thread t2 = new Thread(new MyThread2("B"));
                Thread t3 = new Thread(new MyThread2("C"));

                t1.start();
                t2.start();
                t3.start();
        }
}

class MyThread2 implements Runnable {
        private String name;

        MyThread2(String name) {
                this.name = name;
        }

        @Override
        public void run() {
                // 스레드로 수행될 코드
                for(int i=0; i<10; i++) {
                        System.out.print(name + " ");

                        try {
                                Thread.sleep(10);
                        } catch (InterruptedException e) {
                        }
                }
        }
}
```

join()의 개념 및 실습

아래코드에서 th.start() thread의 작업이 끝날 때 까지 기다린다는 의미로 th.join()을 사용한다.

아래 노란색 블록을 막았을 때와 그렇지 않았을 때의 결과를 비교해 보시오.

```
public class CountingThreadConsole {

        static public void main(String [] args) {
                Thread th = new Thread(new CountingThread());
                th.start();
                System.out.println("After Starting Thread");
                try {
                        th.join();
```

```
                }
                catch(InterruptedException e) {

                }
                System.out.println("After Join");
        }
}

class CountingThread implements Runnable {
        public void run() {
                for(int i=1; i<=100; i++)
                        System.out.println(i);
        }
}
```

실습예제3

이번 예제는 은행에 계좌를 1개 만들고 한사람만 입출금을 1000번 반복했을 경우 잔액이 0 이 된다. 실습해서 확인해 보세요

```
public class ThreadEx3 {
        public static void main(String[] args) throws InterruptedException {
                Account account = new Account();

                BankWork b1 = new BankWork(account); // 은행에 계좌 등록
                b1.start();

            b1.join(); // b1 쓰레드가 종료될 때까지 메인 쓰레드를 기다리게 함.

                System.out.println("잔액: " + account.getBalance());
        }
}

class BankWork extends Thread {
        private Account account;

        BankWork(Account account) {
                this.account = account;
        }

        @Override
        public void run() {
            for(int i=1; i<=10000; i++) {
                account.deposit(i); // 입금
                account.withdraw(i); // 출금
            }
        }

}

class Account {
        private long balance;

        public long getBalance() {
            return balance;
        }

        public void setBalance(long balance) {
            this.balance = balance;
        }
```

```
	public void deposit(int amount) {
		balance = balance + amount;
	}

	public void withdraw(int amount) {
		balance = balance - amount;
	}
}
```

3명이 한 계좌를 동시에 입출금 하는 경우 동기화를 하지 않으면 정상적으로 잔고계산이 되지 않기 때문에 synchronized를 사용한다. 하지만 사용시 수행시간이 오래 걸리는 단점이 있다. 한번 측정해 보자. synchronized를 넣고 빼고 실행 시켜 보고 시간을 비교해 보자.

```
public class ThreadEx3 {
	public static void main(String[] args) throws InterruptedException {
		Account account = new Account();

		BankWork b1 = new BankWork(account); // 은행에 계좌 등록
		BankWork b2 = new BankWork(account); // 은행에 계좌 등록
		BankWork b3 = new BankWork(account); // 은행에 계좌 등록

		b1.start();
		b2.start();
		b3.start();

		// synchronized 를 사용하여 동기화를 하면 수행 시간이 오래 걸린다. 측정해보자
		long from = System.currentTimeMillis();

		b1.join(); // b1 쓰레드가 종료될 때까지 메인 쓰레드를 기다리게 함.
		b2.join();
		b3.join();

		long to = System.currentTimeMillis();

		System.out.println("잔액: " + account.getBalance());
		System.out.println("경과시간: " + (to - from) + "msec");
	}
}

class BankWork extends Thread {
	private Account account;

	BankWork(Account account) {
		this.account = account;
	}

	@Override
	public void run() {
		for(int i=1; i<=10000; i++) {
			account.deposit(i); // 입금
			account.withdraw(i); // 출금
		}
	}

}

class Account {
```

```
	private long balance;

	public long getBalance() {
		return balance;
	}

	public void setBalance(long balance) {
		this.balance = balance;
	}

	// synchronized 를 사용하면 동기화가 된다. 단점은 수행속도가 떨어짐
	public synchronized void deposit(int amount) {
		balance = balance + amount;
	}

	public synchronized void withdraw(int amount) {
		balance = balance - amount;
	}
}
```

비동기 방식 멀티 쓰레드

실습예제4

쓰레드를 사용하지 않을 경우 500의 합, 300의 합, 100의 합이 차례대로 구해진다.

```
public class ThreadEx4 {
	public static void main(String[] args) {
		MyUtil u1 = new MyUtil(500);
		System.out.println(u1.doLongWork());

		MyUtil u2 = new MyUtil(300);
		System.out.println(u2.doLongWork());

		MyUtil u3 = new MyUtil(100);
		System.out.println(u3.doLongWork());
	}
}


class MyUtil {
	private int iteration;

	MyUtil(int iteration) {
		this.iteration = iteration;
	}

	public int doLongWork() {
		int sum = 0;

		for(int i=1; i<=iteration; i++) {
			sum += i;

			try {
				Thread.sleep(20);
			} catch (InterruptedException e) {
			}
		}

		return sum;
	}
```

```
}
```

쓰레드를 생성시켜 실행 시키면 100의 합, 300의 합, 500의 합이 차례대로 출력된다.

```
public class ThreadEx5 {
	public static void main(String[] args) {
		MyUtil2 u1 = new MyUtil2(500, new MyUtil2.ResultHandler() { // 익명의 클래스

			@Override
			public void complete(int result) {
				System.out.println("합계: " + result);
			}
		});

		MyUtil2 u2 = new MyUtil2(300, new MyUtil2.ResultHandler() {

			@Override
			public void complete(int result) {
				System.out.println("합계: " + result);
			}
		});

		MyUtil2 u3 = new MyUtil2(100, (result) -> { System.out.println("합계: " + result); });

		u1.execute();
		u2.execute();
		u3.execute();
	}
}

class MyUtil2 extends Thread {
	public interface ResultHandler {
		void complete(int result);
	}

	private int iteration;
	ResultHandler handler;

	MyUtil2(int iteration, ResultHandler handler) {
		this.iteration = iteration;
		this.handler = handler;
	}

	public void execute() {
		this.start();
	}

	@Override
	public void run() {
		int sum = doLongWork();
		handler.complete(sum);
	}

	public int doLongWork() {
		int sum = 0;
```

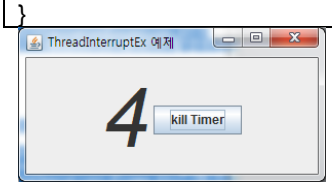
```
                    for(int i=1; i<=iteration; i++) {
                            sum += i;

                            try {
                                    Thread.sleep(20);
                            } catch (InterruptedException e) {
                            }
                    }

                    return sum;
            }
}
```

실습예제5

쓰레드 인터럽트를 통해 실행을 중지시키기

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class TimerRunnable implements Runnable {
        JLabel timerLabel;// 타이머 값이 출력된 레이블

        public TimerRunnable(JLabel timerLabel) {
                this.timerLabel = timerLabel;
        }

        // 스레드 코드.
        // run()이 종료하면 스레드 종료
        public void run() {
                int n=0; // 타이머 카운트 값
                while(true) { // 무한 루프
                        timerLabel.setText(Integer.toString(n)); // 타이머 값을 레이블에 출력
                        n++; // 카운트 증가
                        try {
                                Thread.sleep(1000); // 1초동안 잠을 잔다.
                        }
                        catch(InterruptedException e) {
                                return; // 예외가 발생하면 스레드 종료
                        }
                }
        }
}

public class ThreadInterruptEx extends JFrame {
        Thread th;
        public ThreadInterruptEx() {
                setTitle("ThreadInterruptEx 예제");
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                Container c = getContentPane();
                c.setLayout(new FlowLayout());

                // 타이머 값을 출력할 레이블 생성
                JLabel timerLabel = new JLabel();
                timerLabel.setFont(new Font("Gothic", Font.ITALIC, 80));
```

```
                    // 타이머 스레드로 사용할 Runnable 객체 생성.
                    // 타이머 값을 출력할 레이블 컴포넌트를 생성자에 전달한다.
                    TimerRunnable runnable = new TimerRunnable(timerLabel);
                    th = new Thread(runnable); // 스레드 생성
                    c.add(timerLabel);

                    // 버튼을 생성하고 Action 리스너 등록
                    JButton btn =new JButton("kill Timer");
                    btn.addActionListener(new ActionListener() {
                              public void actionPerformed(ActionEvent e) {
                                        th.interrupt(); // 타이머 스레드 강제 종료
                                        JButton btn = (JButton)e.getSource();
                                        btn.setEnabled(false); // 버튼 비활성화
                              }
                    });
                    c.add(btn);

                    setSize(300,150);
                    setVisible(true);

                    th.start(); // 스레드 시작시킴
          }
          public static void main(String[] args) {
                    new ThreadInterruptEx();
          }
}
```



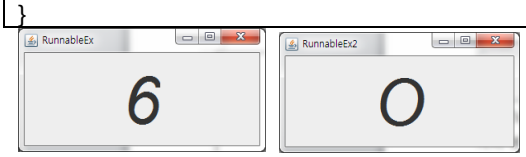
실습예제6

쓰레드를 이용해 카운터 하기

```
import java.awt.*;
import javax.swing.*;

class RunnableTimer2 implements Runnable{
          JLabel timerLabel;
          public RunnableTimer2(JLabel timerLabel2){
                    this.timerLabel=timerLabel2;
          }

          public void run(){
                    char n='A';
                    while(true){
                              timerLabel.setText(Character.toString(n));
                              n++;
                              try {
                                        Thread.sleep(2000);
                              } catch (InterruptedException e) {
                                        return;
                              }
                    }
          }
}

class RunnableTimer implements Runnable{
```

```
	JLabel timerLabel;

	public RunnableTimer(JLabel timerLabel){
		this.timerLabel=timerLabel;
	}

	public void run(){
		int n=0;
		while(true){
			timerLabel.setText(Integer.toString(n));
			n++;
			try {
				Thread.sleep(2000);
			} catch (InterruptedException e) {
				return;
			}
		}
	}
}

class RunnableEx2 extends JFrame{
	public RunnableEx2(){
		setTitle("RunnableEx2");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		Container c=getContentPane();
		c.setLayout(new FlowLayout());

		JLabel timerLabel2=new JLabel();
		timerLabel2.setFont(new Font("Gothic",Font.ITALIC,80));

		RunnableTimer2 runnable=new RunnableTimer2(timerLabel2);

	Thread th=new Thread(runnable);

		c.add(timerLabel2);
		setSize(300,150);
		setVisible(true);
		th.start();
	}
}

public class RunnableEx extends JFrame{
	public RunnableEx(){
		setTitle("RunnableEx");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		Container c=getContentPane();
		c.setLayout(new FlowLayout());

		JLabel timerLabel=new JLabel();
		timerLabel.setFont(new Font("Gothic",Font.ITALIC,80));

	RunnableTimer runnable=new RunnableTimer(timerLabel);

		Thread th=new Thread(runnable);
		c.add(timerLabel);
		setSize(300,150);
		setVisible(true);
		th.start();
	}

	public static void main(String[] args) {
		new RunnableEx();
		new RunnableEx2();
	}
```

```
}
```

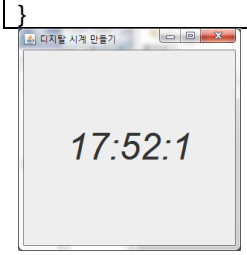


실습예제7

쓰레드를 이용하여 디지털시계 만들기

```
import javax.swing.*;
import java.awt.*;
import java.util.Calendar;

public class DigitalClockFrame extends JFrame {
	public DigitalClockFrame() {
		setTitle("디지탈 시계 만들기");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		add(new MyLabel());
		setSize(300,300);
		setVisible(true);
	}

	class MyLabel extends JLabel implements Runnable {
		boolean bPaused = true;
		Thread timerThread = null;
		public MyLabel() {
			setText(makeClockText());
			setFont(new Font("TimesRoman", Font.ITALIC, 50));
			setHorizontalAlignment(JLabel.CENTER);
			timerThread = new Thread(MyLabel.this);
			timerThread.start();
		}

		public String makeClockText() {
			Calendar c = Calendar.getInstance();
			int hour = c.get(Calendar.HOUR_OF_DAY);
			int min = c.get(Calendar.MINUTE);
			int second = c.get(Calendar.SECOND);

			String clockText = Integer.toString(hour);
			clockText = clockText.concat(":");
			clockText = clockText.concat(Integer.toString(min));
			clockText = clockText.concat(":");
			clockText = clockText.concat(Integer.toString(second));

			return clockText;
		}
		public void run() {
			while(true) {
				try {
					Thread.sleep(1000);
				}
				catch(InterruptedException e){return;}
				setText(makeClockText());
			}
		}
	}

	public static void main(String [] args) {
		new DigitalClockFrame();
```

```
	}
}
```



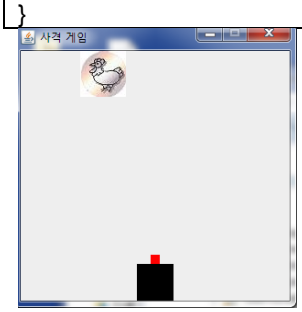
실습예제8

쓰레드를 이용하여 총을 쏘아 치킨 맞추기 게임 실습. 치킨 이미지를 가지고 와야 함.

```
import java.awt.*;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

import javax.swing.*;

public class BulletGameFrame extends JFrame{
	public BulletGameFrame() {
		setTitle("사격 게임");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		GamePanel p = new GamePanel();
		setContentPane(p);
		setSize(300,300);
		setResizable(false);
		setVisible(true);
		p.startGame();
	}

	public static void main(String [] args) {
		new BulletGameFrame();
	}
}
class GamePanel extends JPanel {
	TargetThread targetThread=null;
	JLabel baseLabel = new JLabel();
	JLabel bulletLabel = new JLabel();
	JLabel targetLabel;

	public GamePanel() {
		setLayout(null);

		baseLabel.setSize(40,40);
		baseLabel.setOpaque(true);
		baseLabel.setBackground(Color.BLACK);

		ImageIcon img = new ImageIcon("images/chicken.jpg");
		targetLabel = new JLabel(img);
		targetLabel.setSize(img.getIconWidth(),img.getIconWidth());

		bulletLabel.setSize(10,10);
		bulletLabel.setOpaque(true);
		bulletLabel.setBackground(Color.RED);
		add(baseLabel);
		add(targetLabel);
		add(bulletLabel);
	}
```

```
	public void startGame() {
		baseLabel.setLocation(this.getWidth()/2-20, this.getHeight()-40);
		bulletLabel.setLocation(this.getWidth()/2 - 5, this.getHeight()-50);
		targetLabel.setLocation(0, 0);

		targetThread = new TargetThread(targetLabel);
		targetThread.start();

		baseLabel.requestFocus();
		baseLabel.addKeyListener(new KeyAdapter() {
			BulletThread	bulletThread = null;
			public void keyPressed(KeyEvent e) {
				if(e.getKeyChar() == '\n') {
					if(bulletThread==null || !bulletThread.isAlive()) {
						bulletThread	=	new	BulletThread(bulletLabel,
targetLabel, targetThread);
						bulletThread.start();
					}
				}
			}
		});
	}

	class TargetThread extends Thread {
		JComponent target;
		public TargetThread(JComponent target) {
			this.target = target;
			target.setLocation(0, 0);
			target.getParent().repaint();
		}
		public void run() {
			while(true) {
				int x = target.getX()+5;
				int y = target.getY();
				if(x > GamePanel.this.getWidth())
					target.setLocation(0,0);
				else
					target.setLocation(x, y);

				target.getParent().repaint();
				try {
					sleep(20);
				}
				catch(InterruptedException e) {
					// the case of hit by a bullet
					target.setLocation(0, 0);
					target.getParent().repaint();
					try {
						sleep(500); // 0.5초 기다린 후에 계속한다.
					}catch(InterruptedException e2) {}

				}
			}
		}
	}

	class BulletThread extends Thread {
		JComponent bullet, target;
		Thread targetThread;
		public BulletThread(JComponent bullet, JComponent target, Thread targetThread) {
			this.bullet = bullet;
			this.target = target;
			this.targetThread = targetThread;
		}
```

```
                public void run() {
                        while(true) {
                                // 명중하였는지 확인
                                if(hit()) {
                                        targetThread.interrupt();
                                        bullet.setLocation(bullet.getParent().getWidth()/2 - 5,
bullet.getParent().getHeight()-50);
                                        return;
                                }
                                else {
                                        int x = bullet.getX() ;
                                        int y = bullet.getY() - 5;
                                        if(y < 0) {
                                                bullet.setLocation(bullet.getParent().getWidth()/2 - 5,
bullet.getParent().getHeight()-50);
                                                bullet.getParent().repaint();
                                                return; // thread ends
                                        }
                                        bullet.setLocation(x, y);
                                        bullet.getParent().repaint();
                                }
                                try {
                                        sleep(20);
                                }
                                catch(InterruptedException e) {}
                        }
                }

                private boolean hit() {
                        if(targetContains(bullet.getX(), bullet.getY()) ||
                                targetContains(bullet.getX() + bullet.getWidth() - 1, bullet.getY()) ||
                                targetContains(bullet.getX()+bullet.getWidth()-1,
bullet.getY()+bullet.getHeight() - 1) || targetContains(bullet.getX(), bullet.getY()+bullet.getHeight() - 1))
                                return true;
                        else
                                return false;
                }

                private boolean targetContains(int x, int y) {
                        if(((target.getX() <= x) && (target.getX() + target.getWidth() - 1 >= x)) &&
                           ((target.getY() <= y)&& (target.getY() + target.getHeight() - 1 >= y))) {
                                return true;
                        }
                        else
                                return false;

                }
        }
}
```



수고했습니다. 쓰레드는 다른 기능과 함께 사용되는 경우가 대부분입니다. 쓰레드를 생성시키는 2 가지 방법을 잘 익혀 두었다가 사용해 보도록 하세요.


THINKING CODING