

```

    }

    public synchronized void put(int data) {
        this.data = data;
    }
}

```

실행결과

생산자: 0번케익을생산하였습니다.
 소비자: 0번케익을소비하였습니다.
 소비자: 0번케익을소비하였습니다. ← 똑같은 케익을 3번이나 가져간다.
 소비자: 0번케익을소비하였습니다.
 생산자: 1번케익을생산하였습니다.
 소비자: 1번케익을소비하였습니다.
 ...

동일한 케익을 여러 번 가져가는 것을 알 수 있다. 또 소비가 되지 않았는데도 케익을 생산하는 것을 알 수 있다.

중간점검

1. wait()와 notifyAll() 메소드는 왜 필요한가?
2. wait()는 어떤 역할을 하는가?
3. notifyAll()는 어떤 역할을 하는가?

LAB

INTRODUCTION TO JAVA PROGRAMMING

1. 스레드와 그래픽을 결합하면 초보적인 게임을 제작할 수 있다. 갤러그와 약간만 비슷한 게임을 작성하여 보자. 3개의 객체가 등장한다. 플레이어, 적, 미사일이다. 플레이어 캐릭터는 화살표 키를 이용하여서 상하좌우로 움직일 수 있다. 적은 가로로 왕복한다. 미사일은 스페이스 키를 누르면 발사된다. 미사일이 적에 맞아도 아무런 일도 일어나지 않는다. 일단은 이 3개의 객체를 하나의 스레드로 움직이는 데만 집중하여 보자.



실행결과



Hint

플레이어, 적, 미사일은 모두 공통적인 특징을 공유하고 있다. 그림 파일을 가지고 있으며 움직인다. 또 자기 자신을 화면에 그려야 한다. 따라서 `GraphicObject`라는 슈퍼 클래스를 작성하고 이것을 상속받아 플레이어, 적, 미사일을 작성한다. 또 2개의 중요한 메소드가 있는데 `update()`는 자신의 위치를 변경하는데 사용되고 `draw()`는 자기 자신을 화면에 그리는 데 사용된다.

```
class GraphicObject {
    BufferedImage img = null;
    int x=0, y=0;

    public GraphicObject(String name) {
        try {
            img = ImageIO.read(new File(name));
        } catch (IOException e) {
            System.out.println(e.getMessage());
            System.exit(0);
        }
    }

    public void update() { }
    public void draw(Graphics g) {
        g.drawImage(img, x, y, null);
    }

    public void keyPressed(KeyEvent event) { }
```

플레이어, 적, 미사일의
공통적인 클래스



```

}

class Missile extends GraphicObject {
    boolean launched = false
    public Missile(String name) {
        super(name);
        y = -200;
    }
    public void update() {
        if( launched ) y -= 1;
        if( y < -100 ) launched = false;
    }
    public void keyPressed(KeyEvent event, int x, int y) {
        if( event.getKeyCode() == KeyEvent.VK_SPACE ){
            launched = true;
            this.x = x;
            this.y = y;
        }
    }
}

```

← 스페이스 키가 눌러지면
미사일 발사되도록 한다.

```

class Enemy extends GraphicObject {
    int dx = -10;
    public Enemy(String name) {
        super(name);
        x = 500;
        y = 0;
    }
    public void update() { x += dx;
        if( x < 0 ) dx = +10; if( x > 500 ) dx = -10; }
}

```

← 적은 가로 방향으로
왔다 갔다한다.

```

class Spaceship extends GraphicObject {
    public Spaceship(String name) {
        super(name);
        x = 150;
        y = 350;
    }
    public void keyPressed(KeyEvent event) {
        if( event.getKeyCode() == KeyEvent.VK_LEFT ){ x -= 10; }
        if( event.getKeyCode() == KeyEvent.VK_RIGHT ){ x += 10; }
        if( event.getKeyCode() == KeyEvent.VK_UP ){ y -= 10; }
        if( event.getKeyCode() == KeyEvent.VK_DOWN ){ y += 10; }
    }
}

```

← 화살표 키로 눌러면 플레이어
우주선을 이동시킨다.

```

class MyPanel extends JPanel implements KeyListener {
    Enemy enemy;
    Spaceship spaceship;
    Missile missile;
    public MyPanel() {
        super();
        this.addKeyListener(this);
    }
}

```

레드를 정의하여서 주기적으로 그래픽 객체들의 위치를 업데이트하고 화면에 그림을 그린다.

화면에 그림을 그린다.

키보드 이벤트를 전달한다.

```

    this.requestFocus();
    setFocusable(true);
    enemy = new Enemy("enemy.png");
    spaceship = new SpaceShip("spaceship.png");
    missile = new Missile("missile.png");
    class MyThread extends Thread {
        public void run(){
            while(true){
                enemy.update();
                spaceship.update();
                missile.update();
                repaint();
                try { Thread.sleep(50); }
                catch (InterruptedException e) {}
            }
        }
    }
    Thread t = new MyThread();
    t.start();
}

public void paint(Graphics g) {
    super.paint(g);
    enemy.draw(g);
    spaceship.draw(g);
    missile.draw(g);
}

public void keyPressed(KeyEvent event) {
    spaceship.keyPressed(event);
    missile.keyPressed(event, spaceship.x, spaceship.y);
}

public void keyReleased(KeyEvent arg0) {}
public void keyTyped(KeyEvent arg0) {}
}

public class MyFrame extends JFrame {
    public MyFrame() {
        setTitle("My Game");
        add(new MyPanel());
        setSize(500, 500);
        setVisible(true);
    }

    public static void main(String[] args) {
        new MyFrame();
    }
}

```

도전과제

1. 적 캐릭터가 미사일에 맞으면 소멸되도록 코드를 추가하라.
2. 난수를 발생하여서 적 캐릭터가 움직이는 경로를 불규칙하게 하라.
3. ArrayList를 이용하여서 여러 개의 적 캐릭터를 생성하고 관리하라.

