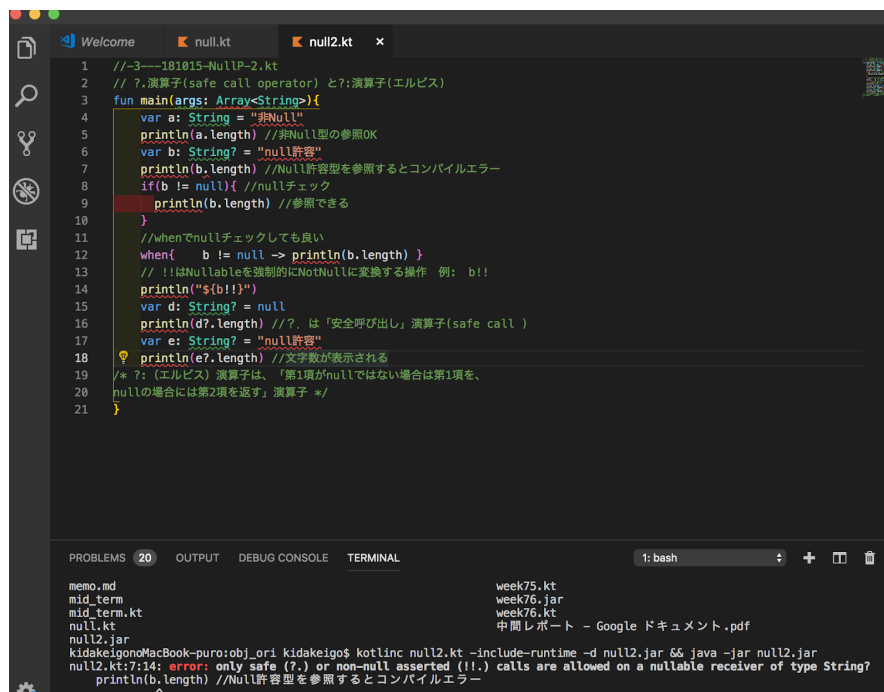


192010  
喜田圭伍  
オブジェクト指向

FR-1 Kotlinのnull許容，非許容を学んで報告する。

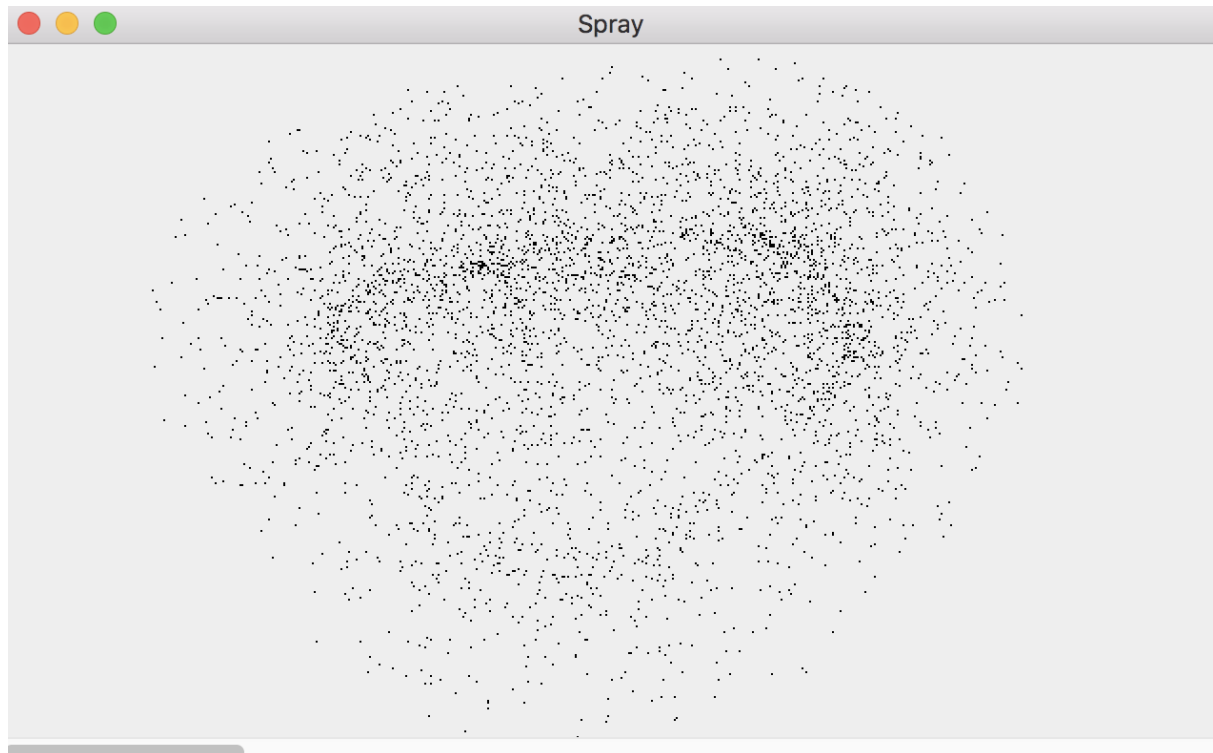


```
1 //3---181015-NullP-2.kt
2 // ?.演算子(safe call operator) と?:演算子(エルビス)
3 fun main(args: Array<String>){
4     var a: String = "非Null"
5     println(a.length) //非Null型の参照OK
6     var b: String? = "null許容"
7     println(b.length) //Null許容型を参照するとコンパイルエラー
8     if(b != null){ //nullチェック
9         println(b.length) //参照できる
10    }
11    //whenでnullチェックしても良い
12    when{ b != null -> println(b.length) }
13    // !!はNullableを強制的にNotNullに変換する操作 例: b!!
14    println("$b!!")
15    var d: String? = null
16    println(d?.length) //?. は「安全呼び出し」演算子(safe call )
17    var e: String? = "null許容"
18    println(e?.length) //文字数が表示される
19    /* ?: (エルビス) 演算子は、「第1項がnullではない場合は第1項を、
20     nullの場合には第2項を返す」演算子 */
21 }
```

String?型にはnullを代入できる。null許容。String型はnullを代入できない。非null許容。  
null許容型のプロパティを参照しようとするとうのようにエラーが出る。nullの可能性があ  
るオブジェクトのプロパティを参照しようとするものが存在しないものにアクセスすることにな  
るためである。if文でnullではないことを確かめてから参照しようすると正常に通る。

・REPLとは対話型評価環境である。試しにプログラムを描いてみたい場合にREPLは有用  
である。

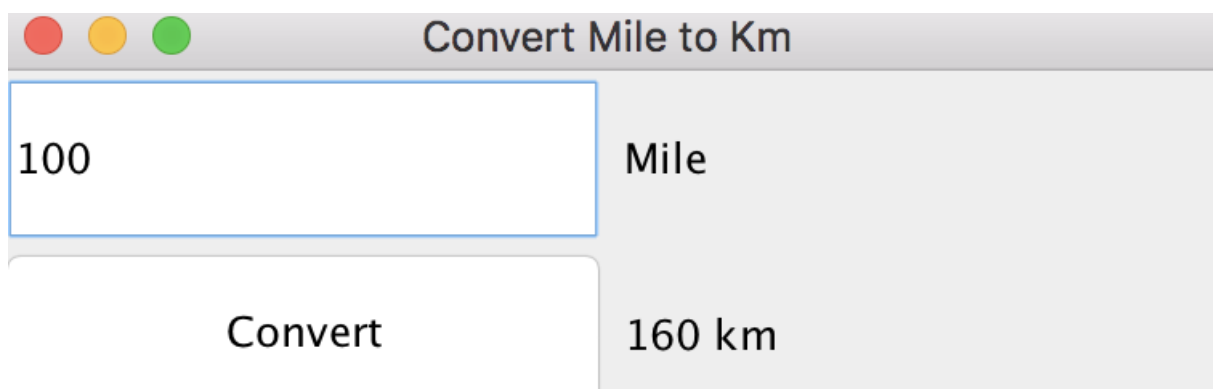
**FR-2 自由な描画プログラムコードを参照して、MouseMotionイベント処理を学ぶ。**



初期値を100にして霧吹きでアートがかけられるようなものにした。

乱数とは次にどのような値が出るかわからないランダムな数のことである。プログラミングにおいてはパスワードの暗号化などセキュリティを高めるために利用されることが多い。またゲームアプリにおいてもゲーム性を高めるために必須である。擬似乱数は高速で低コストであるが周期が短かったり、偶数と奇数が交互に出るものがあるなど問題点がある。メルセンヌ・ツイスター法という対策法が作られた。

**FR-3 度量衡の変換GUIプログラムを作成する**



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

```

fun createAndShowGUI() {
    JFrame.setDefaultLookAndFeelDecorated(true)
    //val converter = CelsiusConverter()
    MileConverter()
}
//
fun main(args:Array<String>) {

    javax.swing.SwingUtilities.invokeLater(object:Runnable {
        public override fun run() {
            createAndShowGUI()
        }
    })
}
//
class MileConverter:ActionListener {
    internal var converterFrame:JFrame? = null
    internal var converterPanel:JPanel? = null
    internal var tempMile:JTextField? = null
    internal var mileLabel:JLabel? = null
    internal var kmLabel:JLabel? = null
    internal var convertTemp:JButton? = null
    init{

        converterFrame = JFrame("Convert Mile to Km")
        converterFrame!!.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
        converterFrame!!.setSize(Dimension(120, 40))

        converterPanel = JPanel(GridLayout(2, 2))

        addWidgets()
        //Set the default button.
        converterFrame!!.getRootPane()!!.setDefaultButton(convertTemp)
        //Add the panel to the window.
        converterFrame!!.getContentPane()!!.add(converterPanel, BorderLayout.CENTER)
        //Display the window.
        converterFrame!!.pack()
        converterFrame!!.setVisible(true)
    }
}
/**
 * Create and add the widgets.

```

```

*/
private fun addWidgets() {
    //Create widgets.
    tempMile = JTextField(2)
    mileLabel = JLabel("Mile", SwingConstants.LEFT)
    convertTemp = JButton("Convert")
    kmLabel = JLabel("Km", SwingConstants.LEFT)
    //Listen to events from the Convert button.
    convertTemp!!.addActionListener(this)
    //Add the widgets to the container.
    converterPanel!!.add(tempMile)
    converterPanel!!.add(mileLabel)
    converterPanel!!.add(convertTemp)
    converterPanel!!.add(kmLabel)
    mileLabel!!.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5))
    kmLabel!!.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5))
}

public override fun actionPerformed(event:ActionEvent?) {
    //Parse degrees Mile as a double and convert to Km.
    val tempKm = (((java.lang.Double.parseDouble(tempMile!!.getText())) / 0.62137)
    ).toInt()
    kmLabel!!.setText(tempKm.toString() + " km")
}

}

//-----

```

## FR-4 マルチスレッド(並列処理)を学ぶ

```

// MovingBall.kt
import java.awt.*
import java.awt.event.*

class Movingball :Frame(), Runnable {
    internal var x = 150
    internal var y = 140
    internal var r = 15
    internal var dx = 8
    internal var dy = 5
    internal var width = 320
    internal var height = 200

```

```

internal var anim:Thread? = null
internal var mvb:Movingball? = null
internal var msw:Boolean = false

init{
    setAnim()
}

internal fun setAnim() {
    //if ((msw == true){
    //if (anim == null)
    //{
        anim = Thread(this)
        anim!!.start()
    }
    //}
    //}

    internal fun endAnim() {
        // anim.stop();
        anim = null
    }

    /* public void start() { anim = new Thread(this); anim.start(); }
    */

    public override fun run() {
        while (anim != null)
        {
            if ((x - r + dx < 0) || (x + r + dx > width))
                dx = -dx
            if ((y - r + dy < 0) || (y + r + dy > height))
                dy = -dy
            x += dx
            y += dy
            repaint()
            try
            {
                Thread.sleep(75)
                //anim!!.stop()
            }
            catch (e:InterruptedException) {}
        }

        public override fun paint(g:Graphics?) {
            // g.setColor(Color.red);
            // g.fillRect(30, 30, 150, 150);

```

```
g!!.fillOval(x - r, y - r, r * 2, r * 2)
}
}

fun main(arg:Array<String>){
    val mvb = Movingball()
    mvb.setSize(320, 200)
    mvb.setBackground(Color.lightGray)
    mvb.setForeground(Color.green)
    mvb.setVisible(true)
}
```

## 参考文献

<http://www.math.sci.hiroshima-u.ac.jp/~m-mat/TEACH/ichimura-sho-koen.pdf>