作成者:　　mitsuya.85u@ray.ocn.ne.jp

6.1. スレッド対応（インタラプト処理）

スロットマシンプログラムをスレッド対応で作成します。

【仕様】
①通常のシングルスレッド処理で作成します。
　まずは３つの数字の回転表示を行い10000回の回転後に状況を表示するプログラムを作成します。
　最初の持ち点は100点、1回ごとの掛け点は10点に固定です。
　３つの数字がマッチすると掛け点の10倍が戻されます。メッセージとして「VeryGood」を返します。
　2つの数字がマッチすると掛け点の2倍が戻されます。メッセージとして「Good」を返します。
　バラバラの数字では戻される点数はありません。メッセージも返しません。
　持ち点がなくなると処理を終了します。

②スレッド・インタラプト処理に変更します。
キーボードからのリターンキーの入力で回転表示を止めるように変更します。

③インタラプト時の処理を追加します。
回転表示が止まっている間に掛け点を再設定できるように変更します。

プログラムはThreadクラス拡張版で実装します。

【実行例】

```
853

別のスレッドから割り込まれました
8:5:3 90

[return]のみで再開します。
掛け点再設定は10秒以内に入力してください
掛け点の入力 1=10点,2=20点,2=20点,3=30点,4=40点,5=50点,6=60点,7=70点,8=80
点,9=90点,0＝100点  ? 3

899

別のスレッドから割り込まれました
8:9:9 150 Good
```

仕様①の作成

【クラス図】

クラス名
Thread

↑

クラス名
SlotThread

フィールド定義
-numA:int
-numB:int
-numC:int
-kakeTen:int
-zanTen:int
-message:String

メソッド定義
+SlotThread():
+run():void
+rolling():void
+result():void
+calc():void
+setKakeTen(k:int):void
+setZanTen(z:int):void

#:protected
-:private
+:public

【実行クラス実装】

●SlotThreadMain.java

```
public class SlotThreadMain{↓
    public static void main(String[] args){↓
        SlotThread st = new SlotThread();↓
        st.setKakeTen(10);↓
        st.setZanTen(100);↓
        st.start();↓
    }↓
}↓
```

【ロジック実装】

●SlotThread.java

```
public void run(){↓
     int i=0;↓
     while(true){↓
           rolling();↓
           if(i==10000){↓
             calc();↓
             result();↓
             i=0;↓
             if(zanTen<=0){↓
                 System.exit(0);↓
             }↓
           }↓
           i++;↓
   }↓
}↓
public void rolling(){↓
           numA=new java.util.Random().nextInt(10)%10;↓
           numB=new java.util.Random().nextInt(10)%10;↓
           numC=new java.util.Random().nextInt(10)%10;↓
           System.out.printf("¥b¥b¥b%d%d%d",numA,numB,numC);↓
}↓
public void result(){↓
           System.out.print(" "+numA+":");↓
           System.out.print(numB+":");↓
           System.out.print(numC+" ");↓
           System.out.print(zanTen);↓
           System.out.println(message);↓
}↓
public void calc(){↓
   int matchi3=10;↓
   int matchi2=2;↓
   message="";↓
   if((numA==numB)&&(numB==numC)){ ↓
         zanTen=zanTen+kakeTen*matchi3;↓
         message=" VerryGood ";↓
   }else if((numA==numB)||(numB==numC)||(numA==numC)){ ↓
         zanTen=zanTen+kakeTen*matchi2;↓
         message=" Good ";↓
   }else{↓
         zanTen=zanTen-kakeTen;↓
   } ↓
}  ↓
public void setKakeTen(int k) {↓
 this.kakeTen=k;↓
}↓
public void setZanTen(int z) {↓
 this.zanTen=z;↓
}↓
}↓
```

【テスト実行】
ここまででmainクラスを実行すると10000回数字が回転すると持ち点の計算結果が出力され再開します。

仕様②の作成

●SlotThread.javaの変更箇所

```
public void run(){↓
    int i=0;↓
    while(true){↓
      try{                    //koko↓
          rolling();↓
          Thread.sleep(70); //koko インタラプトを受け付ける↓
      } catch (InterruptedException ie) {                                    //koko↓
          System.out.println("¥n別のスレッドから割り込まれました");   //koko↓
          calc();                                                            //koko↓
          result();                                                          //koko↓
          System.out.println("");↓
          if(zanTen<=0){↓
            System.exit(0);↓
          }↓
      }↓
  }↓
}↓
```

●SlotThreadMain.javaの変更箇所

```
public static void main(String[] args){↓
    SlotThread2 st = new SlotThread2();↓
    st.setKakeTen(10);↓
    st.setZanTen(100);↓
    st.start();↓
    while(true){    //koko↓
     String s=new java.util.Scanner(System.in).nextLine(); //koko↓
     if (s.equals(""))[ //koko↓
        st.interrupt(); //koko↓
     ]  //koko↓
    ]   //koko↓
}↓
```

【テスト実行】
ここまででmainクラスを実行するとリターンキーの押下で数字の回転が止まり持ち点の計算結果が出力され再開します。

仕様③の作成

●SlotThread.javaの変更箇所

```
public void run(){↓
    int i=0;↓
    while(true){↓
      try{                    //koko↓
          rolling();↓
          Thread.sleep(70); //koko インタラプトを受け付ける↓
      } catch (InterruptedException ie) {                                    //koko↓
          System.out.println("¥n別のスレッドから割り込まれました");   //koko↓
          calc();                                                            //koko↓
          result();                                                          //koko↓
          System.out.println("");↓
          if(zanTen<=0){↓
            System.exit(0);↓
          }↓
////////////////////////////////////////////////////////////////////////////////↓
          try {↓
            System.out.println("[return]のみで再開します。");↓
            System.out.println("掛け点再設定は10秒以内に入力してください");↓
            System.out.print("掛け点の入力 1=10点,2=20点,2=20点,3=30点,4=40点,");↓
            System.out.print("5=50点,6=60点,7=70点,8=80点,9=90点,0＝100点□? ");↓
            Thread.sleep(10000);↓
          }catch(InterruptedException e){↓
            System.out.println("¥n");↓
//          e.printStackTrace();↓
//          System.out.println("リターンキー連打でここに入る");↓
          }↓
////////////////////////////////////////////////////////////////////////////////↓
      }↓
  }↓
}↓
```

●SlotThreadMain.javaの変更箇所

```
public class SlotThread3Main{
    public static void main(String[] args){
        SlotThread3 st = new SlotThread3();
        st.setKakeTen(10);
        st.setZanTen(100);
        st.start();
        while(true){
         String s=new java.util.Scanner(System.in).nextLine();
         ////////////////////////////////////////////////////////
         if (s.equals("1")){st.setKakeTen(10);}
         if (s.equals("2")){st.setKakeTen(20);}
         if (s.equals("3")){st.setKakeTen(30);}
         if (s.equals("4")){st.setKakeTen(40);}
         if (s.equals("5")){st.setKakeTen(50);}
         if (s.equals("6")){st.setKakeTen(60);}
         if (s.equals("7")){st.setKakeTen(70);}
         if (s.equals("8")){st.setKakeTen(80);}
         if (s.equals("9")){st.setKakeTen(90);}
         if (s.equals("0")){st.setKakeTen(100);}
         ////////////////////////////////////////////////////////
         if (s.equals("")){
            st.interrupt();
         }
        }
    }
}
```

【テスト実行】

これで完成です。

mainクラスを実行するとリターンキーの押下で数字の回転が止まり持ち点の計算結果が出力されます。

同時に掛け点の再入力を促します。

リターンキーの押下だけで前に設定された掛け点で再開します。

掛け点を変更したいときは数字を入力してリターンキーを2回押すとで即時再開します。

（1回押すと10秒待つ必要があります）

なにもしないと10秒後に前に設定された掛け点で再開します。

【参考】
ちなみにRunnableインターフェース実装版は以下のようになります。

●Runnableインターフェース実装版

【ソースコード】SlotRunnable.java

```
public class SlotRunnable implements Runnable{
  int numA,numB,numC;
  int kakekin,zankin;
  String mes;

  int matchi3=5;
  int matchi2=2;

public SlotRunnable(){}
public SlotRunnable(int kakekin,int zankin){
  this.kakekin=kakekin;
  this.zankin=zankin;
}
public void run(){
    while(true){
      try{
          numA=new java.util.Random().nextInt(10)%10;
          numB=new java.util.Random().nextInt(10)%10;
          numC=new java.util.Random().nextInt(10)%10;
          System.out.printf("¥b¥b¥b%d%d%d",numA,numB,numC);
          Thread.sleep(80);
      } catch (InterruptedException ie) {
          System.out.println("¥n別のスレッドから割り込まれました");
          calc();
          System.out.print(numA+":");
          System.out.print(numB+":");
          System.out.print(numC+" ");
          System.out.print(zankin);
          System.out.println(mes);

               if(zankin<=0){
                   System.out.println("残金不足で終了します");
                   break;
               }
               try {
                   System.out.println("[return]又は５秒後に再開始します¥n");
                   Thread.sleep(5000);
               }catch(InterruptedException e){
//                 e.printStackTrace();
               }
      }
  }
}
public void calc(){
  mes="";
  if((numA==numB)&&(numB==numC)){
        zankin=zankin+kakekin*matchi3;
        mes=" VerryGood ";
  }else if((numA==numB)||(numB==numC)||(numA==numC)){
        zankin=zankin+kakekin*matchi2;
        mes=" Good ";
  }else{
        zankin=zankin-kakekin;
  }
}
public int getZankin() {
 return zankin;
}
}
```

【ソースコード】SlotRunnableMain.java

```
public class SlotRunnableMain{
    public static void main(String[] args){
        SlotRunnable st = new SlotRunnable(100,500);
        Thread com = new Thread( st );
        com.start();
        while(st.getZankin()>=0){
         String s=new java.util.Scanner(System.in).nextLine();
         if (s.equals("")){
            com.interrupt();
            if(st.getZankin()<=0){break;}

         }
        }
    }
}

```