Java には大きく分けて２種類のタイマーがある。

```
java.util.Timer
javax.swing.Timer
```

■タイマーを使わずに一定間隔で処理を行う

特徴

・次の処理開始までの時間＝現在の処理にかかった時間＋ウエイト時間
・前の処理の終了から次の処理の開始までの時間が一定
・ゲーム等ではオススメしない。

```
class TimerTest3 implements Runnable {
  long current;
  long old;
  public TimerTest3(){
    Thread th = new Thread(this);
    old = System.currentTimeMillis();
    th.start();
  }

  public void run() {
    while (true) {
      current = System.currentTimeMillis();
      System.out.println(current - old);
      old = current;

      try {
        Thread.sleep((int)(Math.random() * 1500));
      } catch (Exception ex){
      }

      try {
        Thread.sleep(1000);
      } catch (Exception ex){
      }
    }
  }
}
```

■ java.util.Timer を使う

特徴

・色々な設定が出来る
・schedule の場合
  ・実行が遅延した場合、そのあとの実行も遅延されます。
  ・最終的に、実行の頻度は通常、指定した期間の対応する頻度よりも若干遅くなります
  ・アニメーションタスクやユーザの入力に応じて一定の活動が実行されるタスクに適している
  ・個人的にはゲームにオススメ
・scheduleAtFixedRate の場合
  ・実行が遅延した場合、「遅れを取り戻す」ために 2 つ以上の実行が連続して行われます
  ・最終的に実行の頻度は、指定した期間の対応する頻度と同じになります
  ・「絶対」時間を反映する作業を繰り返すのに適しています。

```
class TimerTest2   {
  long current;
  long old;
  public TimerTest2(){
    Timer timer = new Timer(true);
    old = System.currentTimeMillis();
    timer.schedule(new TestTask(this), 1000,1000);
```

```
    // timer.scheduleAtFixedRate(new TestTask(this), 1000,1000);
  }

  public void actionPerformed(ActionEvent e){
  }

  public void process(){
    current = System.currentTimeMillis();
    System.out.println(current - old);
    old = current;

    try {
      Thread.sleep((int)(Math.random() * 1500));
    } catch (Exception ex){
    }
  }

  static class TestTask extends TimerTask {
    TimerTest2 tt;
    public TestTask(TimerTest2 tt){
      this.tt = tt;
    }

    public void run(){
      tt.process();
    }
  }
}
```

■ javax.swing.Timer

特徴

・実行が遅延した場合、次の実行時間まで待ちます。
・あまりオススメしない。

class TimerTest1 implements ActionListener {

```
  long current;
  long old;
  public TimerTest1(){
    Timer timer = new Timer(1000,this);
    old = System.currentTimeMillis();
    timer.start();
  }

  public void actionPerformed(ActionEvent e){
    current = System.currentTimeMillis();
    System.out.println(current - old);
    old = current;

    try {
      Thread.sleep((int)(Math.random() * 1500));
    } catch (Exception ex){
    }
  }
}
```