

This is an example of a stopwatch with the ability to pause/resume and reset.

### Contents

- [1](#)  
[StopWatch.java](#)
- [2](#)  
[StopWatchCanvas.java](#)
- [3](#)  
[StopWatchThread.java](#)
- [4 Screenshots](#)
- [5 Download](#)

### StopWatch.java

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class StopWatch extends MIDlet {

    private Display display;
    private StopWatchCanvas canvas;

    public StopWatch() {
        display=Display.getDisplay(this);
        canvas=new StopWatchCanvas(this);
    }

    public void startApp() {
        display.append(canvas);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

}
```

### StopWatchCanvas.java

```
import javax.microedition.lcdui.*;

public class StopWatchCanvas extends Canvas implements CommandListener {

    private Command exit;
    private Command start;
    private Command pause;
```

## A\_stopwatch\_in\_Java\_ME

```
private Command reset;
private Stopwatch midlet;
private StopwatchThread thread;
private final String startValue="00:00:00";
private Image image=null;

public StopwatchCanvas(StopWatch midlet){
this.midlet=midlet;

    //Start the thread, stopwatch will remain paused
    new StopwatchThread(this);
    thread;

//Create the commands
=new Command("Exit", Command.EXIT, 1);
=new Command("Start", Command.ITEM, 1);
=new Command("Pause", Command.ITEM, 1);
=new Command("Reset", Command.SCREEN, 2);

    addCommand
    addCommand
    setCommandListener
}

public void setImage(Image img) {
this.image=img;
}

protected void paint(Graphics g) {
    if(image!=null)
        //Draw text updated by the thread
        g.drawImage(image, getWidth()/2, getHeight()/2, Graphics.VCENTER|Graphics.HCENTER);
    else {
        //Display a white rectangle and black text
        g.setColor(255,255,255);
        g.fillRect(0, 0, this.getWidth(), this.getHeight());
        g.setColor(0, 0, 0);
        g.setFont (Font.getFont (Font.FACE_SYSTEM, Font.STYLE_BOLD, Font.SIZE_LARGE));
        g.drawString(startValue, this.getWidth()/2, this.getHeight()/2, Graphics.BASELINE|Gra
    }
}

public void commandAction(Command c, Displayable d) {
    if(c==exit) {
        //Stop the thread
        thread.killThread();
        //Exit the midlet
        midlet.notifyDestroyed();
    }
    else if(c==start) {
        //Start or resume the stopwatch
        thread.startWatch();
        removeCommand(start);
        addCommand(pause);
        addCommand(reset);
    }
    else if(c==pause) {
        //Pause the stopwatch
        thread.pauseWatch();
        removeCommand(pause);
        addCommand(start);
        addCommand(reset);
    }
}
```

```

    }
    else {
        //Reset the counter
        if(!thread.isPaused()) {
            thread.pauseWatch();
            thread.resetWatch();
            removeCommand(pause);
            removeCommand(reset);
            addCommand(start);
        }
        else {
            thread.resetWatch();
            removeCommand(reset);
        }
        //Clear the image
        setImage(null);
    }
    repaint();
}
}

```

## StopWatchThread.java

```

import java.util.*;
import javax.microedition.lcdui.*;

public class StopWatchThread extends Thread {

    private StopWatchCanvas canvas;
    boolean paused=true;
    boolean stopped=false;
    private Date startTime, pauseTime;
    private long totaltime=0;
    private Image img;

    public StopWatchThread(StopWatchCanvas canvas) {
this.canvas=canvas;
    }

    //Stops the thread
    public synchronized void killThread() {
        =stopped
    }

    //Determines whether or not the thread is running
    public synchronized boolean threadIsRunning() {
return !stopped;
    }

    //Starts the stopwatch (or resumes after being paused)
    public synchronized void startWatch() {
        if(startTime==null)
            startTime=new Date();
if ( img == null) img = Image.createImage(240, 160);
        =paused
        //Wake up the thread

```

## A\_stopwatch\_in\_Java\_ME

```
        notifyAll
    }

    //Pauses the stopwatch
    public synchronized void pauseWatch() {
        paused=true;
        pauseTime ();
        totalTime =time.getTime()-startTime.getTime()+totaltime;
    //Make startTime null so that on startWatch() we get a new startTime
        startTime
    }

    //Resets the stopwatch
    public synchronized void resetWatch() {
    //Make startTime null
        startTime=null;
    //Reset totaltime
        totalTime
    }

    public synchronized boolean isPaused() {
        return paused;
    }

    private String getTime() {
    Calendar c=Calendar.getInstance();
    Date rightNow=new Date();
        setTime(new Date(rightNow.getTime()-startTime.getTime()+totaltime));

        //We work with minutes, seconds and centiseconds
        int min=c.get(c.MINUTE);
        int sec=c.get(c.SECOND);
        int csec=c.get(c.MILLISECOND)/10;
    String smin;
    String ssec;
        String scsec;

    //Format the values so that they will look appropriate when displayed
    if(min<10)
        smin="0"+min;
        else
            smin=String.valueOf(min);
    if(sec<10)
        ssec="0"+sec;
        else
            ssec=String.valueOf(sec);
        if(csec<10)
            scsec="0"+csec;
        else
            scsec=String.valueOf(csec);

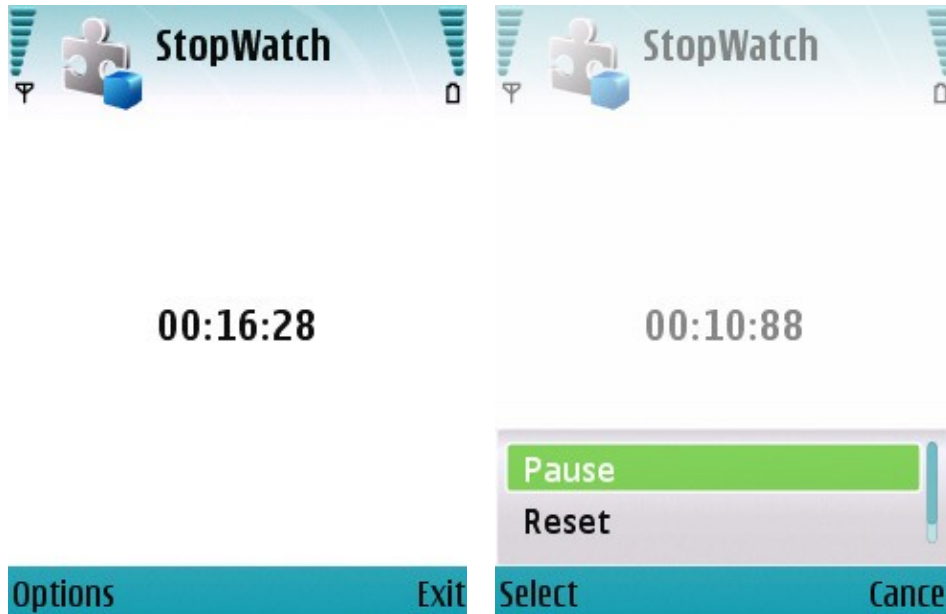
    return smin+": "+ssec+": "+scsec;
    }

    public void run() {
        while(threadIsRunning()) {
            try {
                if(!isPaused()) {
                    try {
    Graphics g=img.getGraphics();
                        setColor(255,255,255);g.
                        fillRect(0, 0, img.getWidth(), img.getHeight());
                    }
                }
            }
        }
    }
}
```

## A\_stopwatch\_in\_Java\_ME

```
setColor(0,0,0);      g.  
setFont(Font.getFont(Font.FACE_SYSTEM, Font.STYLE_BOLD, Font.SIZE_LARGE));  
drawString(getTime(), gmg.getWidth()/2, img.getHeight()/2, Graphics.BASELINE|Graphics.HCENTER);  
    setImage(img);    canvas.  
    repaint();        canvas.  
}  
        catch(Exception e) {  
            }  
            Thread.sleep(10);  
}  
        else {  
            synchronized(this) {  
                wait();  
            }  
}  
    }  
    catch(InterruptedException e) {  
    }  
}  
    }  
}
```

## Screenshots



## Download

[Project containing source code, JAR and JAD](#)