



Here is a sample Midlet: [Media:CustomInputMidlet.zip](#)

Here is a basic example showing how to create a custom text input using J2me and Canvas, things that is often needed when using low level graphics (e.g. for games).

In this code, you can:

- define which characters map to each key
- define blinking interval
- define max interval between subsequent key presses
- move left/right within the text
- delete text

A lot of features are missing, so if you want to implement them you're welcome :)

```
package com.jappit.custominput.screen;

import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Graphics;

public class CustomInputCanvas extends Canvas implements Runnable
{
    static final char[] KEY_NUM1_CHARS = new char[]{'.', '?', '!'};
    static final char[] KEY_NUM2_CHARS = new char[]{'a', 'b', 'c'};
    static final char[] KEY_NUM3_CHARS = new char[]{'d', 'e', 'f'};
    static final char[] KEY_NUM4_CHARS = new char[]{'g', 'h', 'i'};
    static final char[] KEY_NUM5_CHARS = new char[]{'j', 'k', 'l'};
    static final char[] KEY_NUM6_CHARS = new char[]{'m', 'n', 'o'};
    static final char[] KEY_NUM7_CHARS = new char[]{'p', 'q', 'r', 's'};
    static final char[] KEY_NUM8_CHARS = new char[]{'t', 'u', 'v'};
    static final char[] KEY_NUM9_CHARS = new char[]{'w', 'x', 'y', 'z'};
    static final char[] KEY_NUM0_CHARS = new char[]{' '};

    int clearKeyCode = Integer.MIN_VALUE;

    StringBuffer currentText = new StringBuffer();
    String currentString = new String();

    int lastPressedKey = Integer.MIN_VALUE;
    int currentKeyStep = 0;

    Font inputFont = null;
    int inputWidth = 0;
    int inputHeight = 0;
    int inputTranslationX = 0;

    long lastKeyTimestamp = 0;
    long maxKeyDelay = 500L;

    int caretIndex = 0;
    int caretLeft = 0;
    boolean caretBlinkOn = true;
    long caretBlinkDelay = 500L;
    long lastCaretBlink = 0;

    boolean goToNextChar = true;
```

Custom_Text_Input_in_Java_ME

```
public CustomInputCanvas()
{
    new Thread(this).start();

    = Font.getDefaultFont();

    = getWidth();

    = input.getHeight();
}

public char[] getChars(int key)
{
    switch(key)
    {
    case Canvas.KEY_NUM1: return KEY_NUM1_CHARS;
    case Canvas.KEY_NUM2: return KEY_NUM2_CHARS;
    case Canvas.KEY_NUM3: return KEY_NUM3_CHARS;
    case Canvas.KEY_NUM4: return KEY_NUM4_CHARS;
    case Canvas.KEY_NUM5: return KEY_NUM5_CHARS;
    case Canvas.KEY_NUM6: return KEY_NUM6_CHARS;
    case Canvas.KEY_NUM7: return KEY_NUM7_CHARS;
    case Canvas.KEY_NUM8: return KEY_NUM8_CHARS;
    case Canvas.KEY_NUM9: return KEY_NUM9_CHARS;
    case Canvas.KEY_NUM0: return KEY_NUM0_CHARS;
    }
    return null;
}

boolean isClearKey(int key)
{
    return key == -8;
}

void clearChar()
{
    if(currentText.length() > 0 && caretIndex > 0)
    {
        --;          caretIndex

        deleteCharAt(caretIndex);

        = currentText.toString();
    }
}

void updateCaretPosition()
{
    = input.getFontMetrics().stringWidth(currentString, 0, caretIndex);

    if(caretLeft + inputTranslationX < 0)
    {
        = - caretLeftTranslationX
    }
    else if(caretLeft + inputTranslationX > inputWidth)
    {
        = input.getWidth() - caretLeftTranslationX;
    }
}

public void keyPressed(int key)
{
    int gameAction = getGameAction(key);
```

Custom_Text_Input_in_Java_ME

```
System.out.println("KEY: " + key + ", " + gameAction);

if(isClearKey(key))
{
    ();          clearChar

                (); updateCaretPosition

                = true;    goToNextChar
}
else if(key >= KEY_NUM0 && key <= KEY_NUM9)
{
    (key);      writeKeyPressed
}
else if(gameAction == Canvas.LEFT)
{
    if(caretIndex > 0)
    {
        --;          caretIndex

                ();          updateCaretPosition

                = true;          goToNextChar
    }
}
else if(gameAction == Canvas.RIGHT)
{
    if(caretIndex < currentText.length())
    {
        if(goToNextChar)
            ++;          caretIndex

                ();          updateCaretPosition

                = true;          goToNextChar
    }
}
}

public void writeKeyPressed(int key)
{
    if(goToNextChar || key != lastPressedKey)
    {
        = true;    goToNextChar

        = key;    lastPressedKey

        = 0;     currentKeyStep
    }
    else
    {
        ++;     currentKeyStep
    }
}

char[] chars = getChars(key);

if(chars != null)
{
    if(currentKeyStep >= chars.length)
    {
        -= chars.length; currentKeyStep
    }
    if(goToNextChar)
```

Custom_Text_Input_in_Java_ME

```

    insert(caretIndex, characters[currentKeyStep]);
        ++;                caretIndex
}
else
{
    setCharAt(caretIndex, characters[currentKeyStep]);
}
    = currentText.toString();
        (); updateCaretPosition
    = System.currentTimeMillis();
    = false;    goToNextChar
}
}

public void checkTimestamps()
{
    long currentTime = System.currentTimeMillis();

    if(lastCaretBlink + caretBlinkDelay < currentTime)
    {
        = !caretBlinkOn;    caretBlinkOn
        = currentTime;    caretBlink
    }

    if(!goToNextChar && lastKeyTimestamp + maxKeyDelay < currentTime)
    {
        = true;    goToNextChar
    }
}

public void run()
{
    while(true)
    {
        ();    checkTimestamps
        ();    repaint
    }

    try
    {
        synchronized(this)
        {
            (50L);    wait
        }
    }
    catch(Exception e)
    {
        printStackTrace();    e.
    }
}

public void paint(Graphics g)
{
    setFont(inputFont);
}

```

Custom_Text_Input_in_Java_ME

```
setColor(0xffff00ff);  
  
fillRect(0, 0, getWidth(), getHeight());  
  
setColor(0x000000);  
  
translate(inputTranslationX, 0);  
  
drawString(currentString, 0, 0, Graphics.LEFT | Graphics.TOP);  
  
if(caretBlinkOn && goToNextChar)  
{  
    drawLine(caretLeft, 0, caretLeft, inputHeight);  
}  
translate(- inputTranslationX, 0);  
}  
}
```