```java
/*
Calcola il valor medio dei massimi 'blu' di ogni riga
In tutto il programma le eccezioni sono scarsamente gestite:
in genere sono semplicemente intercettate o rilanciate

Programmi più complessi e/o professionali dovrebbero gestire effettivamente le eccezioni
*/

public class AnalisiPerRighe {

    // InterruptedException viene generata da join
    public static void main(String[] args) throws InterruptedException {
        if(args.length==0) {
            System.err.println("uso: java AnalisiPerRighe nome-file-immagine");
            System.exit(1);
        }
        java.awt.image.BufferedImage image = IOTools.read(args[0]);
        int w = image.getWidth();
        int h = image.getHeight();
        Riga[] righe = new Riga[h];
        Thread[] threads = new Thread[h];
        for(int i=0; i<h; i++) {
            int[] rgb = image.getRGB(0, i, w, 1, null, 0, w);
            righe[i] = new Riga(rgb);
            threads[i] = new Thread(righe[i]);
        }
        for(int i=0; i<h; i++) {
            threads[i].start();
        }
```

```
30          for(int i=0; i<h; i++) {
31              threads[i].join(); // *
32              somma += righe[i].getMax();
33          }
34          somma /= h;
35          System.out.println("Blu medio: "+somma);
36      }
37
38      static int somma = 0;
39  }
40
41
42  /*
43  se commento la riga *
44
45  //threads[i].join();
46
47  ottengo
48  Exception in thread "main" java.lang.RuntimeException: done == false
49          at Riga.getMax(Riga.java:29)
50          at AnalisiPerRighe.main(AnalisiPerRighe.java:28)
51          at TestAnalisiPerRighe.main(TestAnalisiPerRighe.java:5)
52  */
53
```

```java
/*
Funzioni utili per l'I/O
in questo caso vi è solo il metodo per leggere una immagine
*/
import java.io.File;
import java.io.IOException;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;

public class IOTools {

    // non fa altro che chiamare il metodo di libreria read e intercettare
    // eventuali eccezioni
    public static BufferedImage read(String fileName) {
        try {
            File f = new File(fileName);
            return ImageIO.read(f);
        } catch(IOException e) {
            e.printStackTrace();
            System.exit(1);
            return null; // altrimenti il compilatore si lamenta
        }
    }

    private IOTools() {} // altrimenti javadoc mostra il costruttore di default
}

```

```java
/*
La classe tratta una riga di una immagine, calcolando il valore massimo di blue presente.
*/

public class Riga implements Runnable {

    private int[] rgbArray;
    private int max = -1;
    private boolean done = false;

    Riga(int[] a) {
        rgbArray = a; //a.clone(); se non sono sicuro di un uso esclusivo di a
    }

    // scandisce la riga  e trova il massimo
    public void run() {
        for(int i=0; i<rgbArray.length; i++) {
            int blu = rgbArray[i] & 255;
            if(blu > max) max = blu;
        }
        done = true;
        // quando ha finito mette il flag a vero
        // rgbArray = null se non mi serve più
    }

    // restituisce il valore massimo trovato
    public int getMax() {
        // se il valore massimo non è ancora stato calcolato
        // genera una eccezione
```

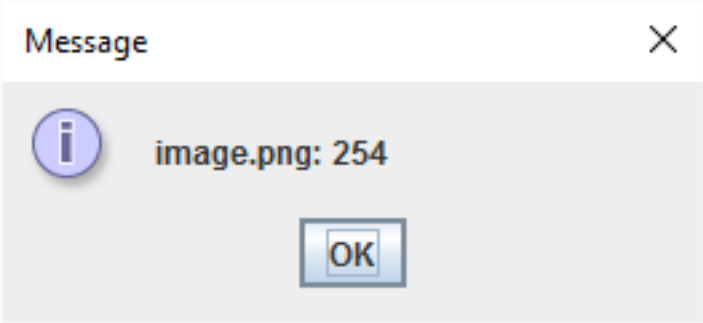
```
30          if(!done) throw new RuntimeException("done == false");
31          return max;
32      }
33
34      /*
35      in alternativa si potrebbe gestire un comportamento "bloccante"
36      ad esempio:
37      while(!done) {
38          // gestisci l'attesa del completamento
39      }
40      */
41
42  }
43
```

```java
/*
Per un uso non interattivo in ambiente grafico
*/

public class TestAnalisiPerRighe {

    public static void main(String[] args) throws InterruptedException {
        // se non sono passati argomenti usa l'immagine image.png
        // nella directory corrente
        if(args.length==0) args = new String[] { "image.png" };
        AnalisiPerRighe.main(args);
        // se vi sono stati problemi durante l'Output
        // viene mostrato il risultato in maniera grafica
        if(System.out.checkError()) {
            javax.swing.JOptionPane.showMessageDialog(
                null, args[0] + ": " + AnalisiPerRighe.somma);
        }
    }

    private TestAnalisiPerRighe() {}
}
```



```
java -jar AnalisiPerRighe.jar %1
pause
```

```
#! /bin/bash

java -jar AnalisiPerRighe.jar $1
read -sn 1 -p "Premi un tasto per continuare"
```

```
%.class: %.java
    javac $<

#viene creato un fila Jar che usa come main la classe TestAnalisiPerRighe
AnalisiPerRighe.jar: *.class Makefile AnalisiPerRighe.bat AnalisiPerRighe.sh
    jar cvfe AnalisiPerRighe.jar TestAnalisiPerRighe *.class *.java AnalisiPerRighe.bat AnalisiPerRighe.sh Makefile

#con il comando make run IF=image.png posso testare il programma
run: AnalisiPerRighe.jar
    java -jar AnalisiPerRighe.jar $(IF)

doc: html/index.html

jar: AnalisiPerRighe.jar

html/index.html: *.java
    javadoc -nodeprecated -nohelp -d html *.java

AnalisiPerRighe.sh:
    echo -e '#! /bin/bash\n\njava -jar AnalisiPerRighe.jar $$1\nread -sn 1 -p "Premi un tasto per continuare"' > $@

AnalisiPerRighe.bat:
    echo -e 'java -jar AnalisiPerRighe.jar %1\npause' > $@

all:
    javac *.java

AnalisiPerRighe.tgz:
    tar -zcvf AnalisiPerRighe.tgz *.java Makefile

scripts: AnalisiPerRighe.sh AnalisiPerRighe.bat
```