Esercitazione 2

Java: Eccezioni

@andreamocci

Casi Eccezionali (1)

```
/*
 * Produce un numero complesso a partire
   dalla sua forma polare
 */
public static Complex
      fromPolarForm(double abs, double phase) {
   /* ricorda : z = modulus * (cos(phase) + i sin(phase)) */
   double re = abs * Math.cos(phase);
   double im = abs * Math.sin(phase);
   return new Complex(re, im);
}
```

Casi Eccezionali (1)

```
/*
 * Produce un numero complesso a partire
   dalla sua forma polare
 */
public static Complex
      fromPolarForm(double abs, double phase) {
   /* ricorda : z = modulus * (cos(phase) + i sin(phase)) */
   double re = abs * Math.cos(phase);
   double im = abs * Math.sin(phase);
   return new Complex(re, im);
}
```

Casi Eccezionali (1)

```
/*
 * Produce un numero complesso a partire
   dalla sua forma polare
 */
public static Complex
      fromPolarForm(double abs, double phase) {
   /* ricorda : z = modulus * (cos(phase) + i sin(phase)) */
   double re = abs * Math.cos(phase);
   double im = abs * Math.sin(phase);
   return new Complex(re, im);
}
```

abs < 0 non e' un valore corretto per il metodo

Casi Eccezionali (1)

```
  /*
   * Produce un numero complesso a partire
     dalla sua forma polare
   */
  public static Complex
        fromPolarForm(double abs, double phase) {
     if (abs < 0) {
       throw new IllegalArgumentException("abs cannot be
negative");
     }
     /* ricorda : z = modulus * (cos(phase) + i sin(phase)) */
     double re = abs * Math.cos(phase);
     double im = abs * Math.sin(phase);
     return new Complex(re, im);
  }
```

Casi Eccezionali

```
  /*
   * Produce un numero complesso a partire
     dalla sua forma polare
   */
  public static Complex
         fromPolarForm(double abs, double phase) {
     if (abs < 0) {
       throw new IllegalArgumentException("abs cannot be
negative");
     }
     /* ricorda : z
     double re = ab
     double im = ab
     return new Co
  }
```

Eccezione Unchecked in java.lang:
"Thrown to indicate that a method has been passed an illegal or inappropriate argument."

Casi Eccezionali (2)

```
// Aggiunge un numero complesso all'insieme.
   public void add(Complex element) {
      boolean containsElement = this.contains(element);

      // se e' gia' presente non faccio nulla
      if (containsElement)
        return;

      /* controllo che il set non sia pieno */
      if (this.size < ComplexSet.MAX_SIZE) {
        this.elements[size] = element;
        this.size++;
      } /* se il set e' pieno, non faccio nulla */
   }
```

Casi Eccezionali (2)

```
// Aggiunge un numero complesso all'insieme.
   public void add(Complex element) {
      boolean containsElement = this.contains(element);

      // se e' gia' presente non faccio nulla
      if (containsElement)
        return;

      /* controllo che il set non sia pieno */
      if (this.size < ComplexSet.MAX_SIZE) {
        this.elements[size] = element;
        this.size++;
      } /* se il set e' pieno, non faccio nulla */
   }
```

vs

Casi Eccezionali (2)

```
// Aggiunge un numero complesso all'insieme.
   public void add(Complex element) {
      boolean containsElement = this.contains(element);

      // se e' gia' presente non faccio nulla
      if (containsElement)
        return;

      /* controllo che il set non sia pieno */
      if (this.size < ComplexSet.MAX_SIZE) {
        this.elements[size] = element;
        this.size++;
      } /* se il set e' pieno, non faccio nulla */
   }
```

vs

Casi Eccezionali (2)

```
// Aggiunge un numero complesso all'insieme.
   public void add(Complex element) {
      boolean containsElement = this.contains(element);

      // se e' gia' presente non faccio nulla
      if (containsElement)
        return;
```

In questo caso l'operazione di aggiunta è definita - (anche) in senso matematico.

E' perfettamente lecito aggiungere un elemento ad un insieme che già lo contiene:

- semplicemente il suo stato non cambia (non rappresenta un insieme diverso)

Casi Eccezionali (2)

In questo caso il componente si trova in uno stato particolare - rispetto al comportamento normale - in cui l'operazione non è lecita.
Non fare nulla (cioè non informare il client) non è in generale una buona pratica

```
    /* controllo che il set non sia pieno */
    if (this.size < ComplexSet.MAX_SIZE) {
      this.elements[size] = element;
      this.size++;
    } /* se il set e' pieno, non faccio nulla */
}
```

Casi Eccezionali (2)

```
// Aggiunge un numero complesso all'insieme.
   public void add(Complex element) {
       ...
       /* controllo che il set non sia pieno */
       if (this.size < ComplexSet.MAX_SIZE) {
          this.elements[size] = element;
          this.size++;
       } /* se il set e' pieno, non faccio nulla */
        else {
          /* che eccezione lanciare? */
       }
   }
```

Casi Eccezionali (2)

```
// Aggiunge un numero complesso all'insieme.
   public void add(Complex element) {
       ...
       /* controllo che il set non sia pieno */
       if (this.size < ComplexSet.MAX_SIZE) {
          this.elements[size] = element;
          this.size++;
       } /* se il set e' pieno, non faccio nulla */
        else {
          /* che eccezione lanciare? Checked vs Unchecked */
       }
   }
```

Unchecked vs Checked

- Eccezioni di tipo aritmetico / logico
- C'è un modo conveniente e poco costoso di evitare l'eccezione
  - Quando il client può controllare che lo stato del componente sia corretto ed evitare lo stato eccezionale

- Ogni qual volta si tratta di un errore la cui causa non è a priori determinabile dal client prima della chiamata (e.g., errori di rete), e a cui il client si deve occupare di porvi rimedio
- Da usare con moderazione

Unchecked

- Eccezioni di tipo aritmetico / logico
- C'è un modo conveniente e poco costoso di evitare l'eccezione
  - Quando il client può controllare che lo stato del componente sia corretto ed evitare lo stato eccezionale

- E' un caso adatto al nostro: è una eccezione di tipo "logico", in quanto il componente si trova in uno stato particolare dove la add non è abilitata.
- Abbiamo due scelte: eccezione già definita nella JDK o una custom

IllegalStateException

```
// Aggiunge un numero complesso all'insieme.
   public void add(Complex element) {
       ...
       /* controllo che il set non sia pieno */
       if (this.size < ComplexSet.MAX_SIZE) {
          this.elements[size] = element;
          this.size++;
       } /* se il set e' pieno, non faccio nulla */
        else {
          throw new IllegalStateException("set is full");
       }
   }
```

IllegalStateException

```
// Aggiunge u
   public voi
       ...
       /* cont
       if (thi
          this.
          this.
       } /* se
        else {
          throw new IllegalStateException("set is full");
       }
   }
```

Eccezione Unchecked in java.lang:
"Signals that a method has been invoked at an illegal or inappropriate time. In other words, the Java environment or Java application is not in an appropriate state for the requested operation."

Eccezione Custom

```
package it.polimi.deib.se.ese1;

public class FullSetException extends RuntimeException {

   ...

   /* implicito: non è necessario dichiararlo */
   public FullSetException() {
      super();
   }

   /* da ridichiarare se serve. Serve? */
   public FullSetException(String message) {
      super(message);
   }
}
```

Eccezione Custom

```
package it.polimi.deib.se.ese1;

public class FullSetException extends RuntimeException {

   ...

   /* implicito: non è necessario dichiararlo */
   public FullSetException() {
      super();
   }

   /* da ridichiarare se serve. Serve? */
   public FullSetException(String message) {
      super(message);
   }
}
```

Soluzione

```
// Aggiunge un numero complesso all'insieme.
   public void add(Complex element) throws
       FullSetException /* meglio dichiararla */ {
       ...
       /* controllo che il set non sia pieno */
       if (this.size < ComplexSet.MAX_SIZE) {
          this.elements[size] = element;
          this.size++;
       } /* se il set e' pieno, non faccio nulla */
       else {
          throw new FullSetException();
       }
   }
```

Problema

- E’ buona norma (si deve) usare una eccezione unchecked quando il client può controllare che lo stato del componente sia corretto ed evitare lo stato eccezionale

Soluzione

```
// restituisce true se e solo se il set e' pieno
public boolean isFull() {
   return this.size == ComplexSet.MAX_SIZE;
}
```

Due Stili

"Ask for forgiveness":

```
try { set.add(new Complex(1.0,1.0)); }
catch (FullSetException e) { /* ... */ }
```

"Ask for permission":

```
if (!set.isFull())
  s.add(new Complex(1.0,1.0));
```

Controllare prima è lo stile tipico in Java (non in altri linguaggi)

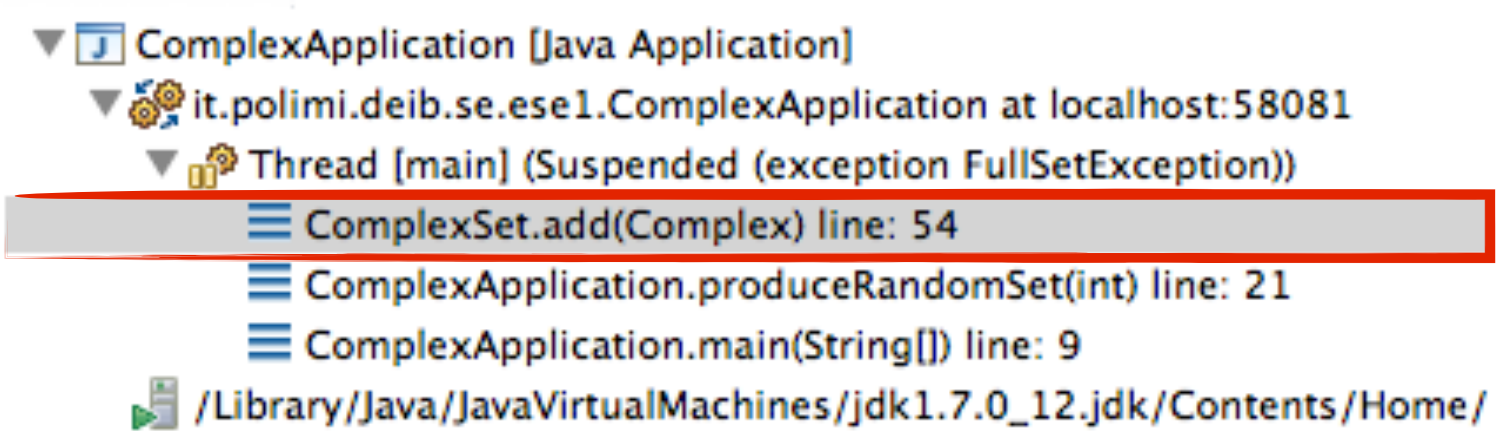
Altro Esempio

```
import java.util.Random;

public class ComplexApplication {
   ...
   public ComplexSet produceRandomSet(int size) {
      ComplexSet set = new ComplexSet();
      Random rand = new Random(System.currentTimeMillis());

      for (int i = 0; i < size; i++) {
         double randRe = rand.nextDouble();
         double randIm = rand.nextDouble();
         Complex randComplex =
              new Complex(randRe, randIm);
         set.add(randComplex);
      }
      return set;
   }
```

Esempio con bug

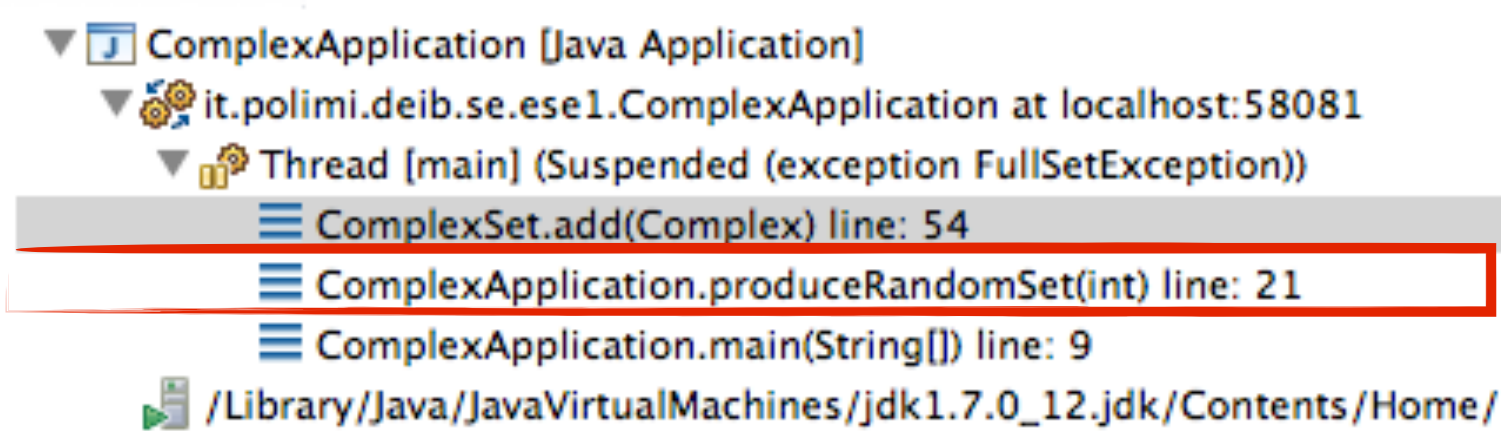
```
import java.util.Random;

public class ComplexApplication {
   ...
   public ComplexSet produceRandomSet(int size) {
      ComplexSet set = new ComplexSet();
      Random rand = new Random(System.currentTimeMillis());

      for (int i = 0; i < size; i++) {
         double randRe = rand.nextDouble();
         double randIm = rand.nextDouble();
         Complex randComplex =
               new Complex(randRe, randIm);
         set.add(randComplex);
      }
      return set;
   }
}
```

Propagazione



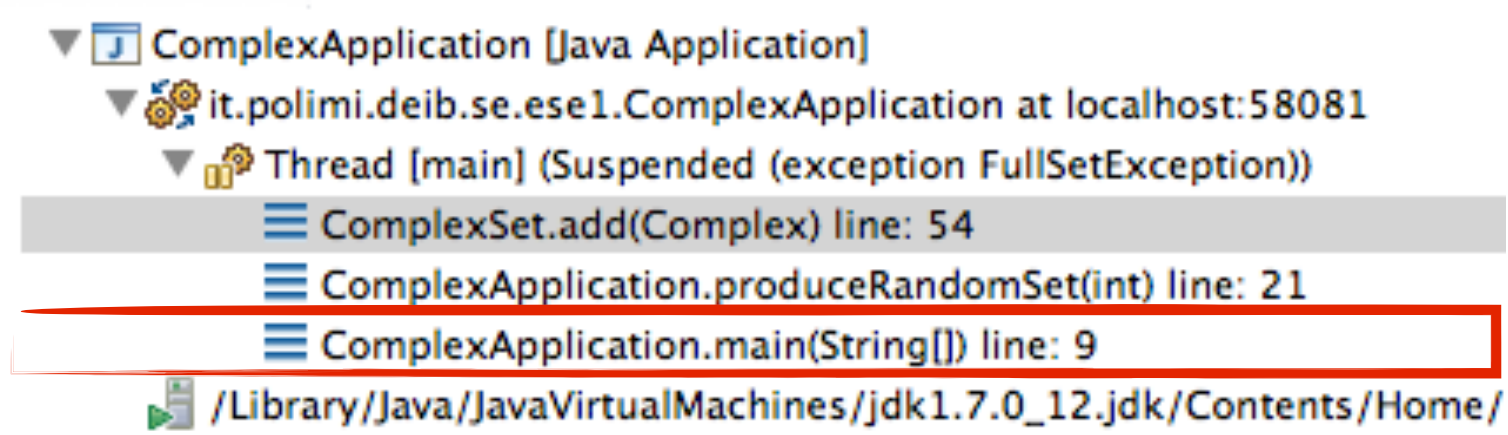

```
if (this.size < ComplexSet.MAX_SIZE) {
        this.elements[size] = element;
        this.size++;
    }
    else {
        throw new FullSetException();
    }
```

Propagazione



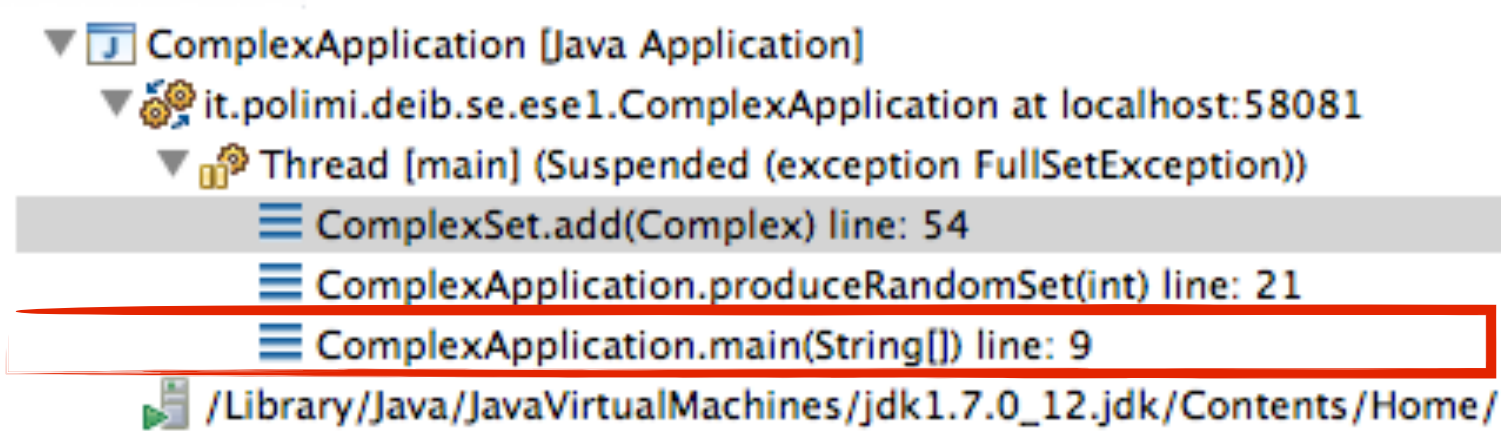

```
        Complex randComplex =
                new Complex(randRe, randIm);
        set.add(randComplex);
    }
    return set;
}
```

Propagazione




```
public static void main(String[] args) {
   ComplexApplication app = new ComplexApplication();
   app.produceRandomSet(101);
}
```

Propagazione




```
Exception in thread "main" it.polimi.deib.se.ese1.FullSetException
   at ComplexSet.add(ComplexSet.java:54)
   at ComplexApplication.produceRandomSet(ComplexApplication.java:21)
   at ComplexApplication.main(ComplexApplication.java:9)
```

Gestione

```
...
for (int i = 0; i < size; i++) {
        double randRe = rand.nextDouble();
        double randIm = rand.nextDouble();
        Complex randComplex =
             new Complex(randRe, randIm);
        try {
           set.add(randComplex);
        } catch (FullSetException e) {
           System.err.println("Errore, set pieno");
           return set; // oppure break;
        }
     }
...
```

Gestione (meglio)

```
...
    for (int i = 0; i < size; i++) {
      double randRe = rand.nextDouble();
      double randIm = rand.nextDouble();
      Complex randComplex =
           new Complex(randRe, randIm);
      if (!set.isFull()) {
        set.add(randComplex);
      } else {
        System.err.println("Errore, set pieno");
        return set; // oppure break;
      }
    }
...
```

Altro esempio

```
public static void writeSet(ComplexSet set, String filename)
     throws IOException {
     FileOutputStream fos = null;
     ObjectOutputStream os = null;

     fos = new FileOutputStream(filename);
     os = new ObjectOutputStream(fos);

     for (Complex complex: set.toArray()) {
        os.writeObject(complex.re());
        os.writeObject(complex.im());
     }
     if (os != null) os.close();
     if (fos != null) fos.close();
  }
```

Molti di questi metodi di classi della JDK possono lanciare eccezioni

Gestione

```
public static void writeSet(ComplexSet set, String filename)
     throws IOException {
     FileOutputStream fos = null;
     ObjectOutputStream os = null;

     fos = new FileOutputStream(filename);
     os = new ObjectOutputStream(fos);

     for (Complex complex: set.toArray()) {
       os.writeObject(complex.re());
       os.writeObject(complex.im());
     }
     if (os != null) os.close();
     if (fos != null) fos.close();
   }
```

FileNotFoundException

Gestione

```
try {
   fos = new FileOutputStream(filename);
} catch (FileNotFoundException e) {
   throw new IllegalArgumentException("Invalid file", e);
}
```

Responsabilità del Client:
Chained (Unchecked) Exception
La causa dell'eccezione è la FileNotFoundException

Gestione

```
public static void writeSet(ComplexSet set, String filename)
     throws IOException {
     FileOutputStream fos = null;
     ObjectOutputStream os = null;

     fos = new FileOutputStream(filename);
     os = new ObjectOutputStream(fos);

     for (Complex complex: set.toArray()) {
        os.writeObject(complex.re());
        os.writeObject(complex.im());
     }
     if (os != null) os.close();
     if (fos != null) fos.close();
  }
```

IOException

Gestione?

- Supponiamo di voler gestire esplicitamente gli errori di I/O in qualche modo
  - Non importa come: chiamiamo un metodo log(...) che fa operazioni di logging
  - Come strutturare il codice?

Gestione

```
try {
   os = new ObjectOutputStream(fos);
   for (Complex complex: set.toArray()) {
      os.writeObject(complex.re());
      os.writeObject(complex.im());
   }
   return;
} catch (IOException ex) {
   log(fos,os,ex);
   return;
} finally {
   if (os != null) os.close();
   if (fos != null) fos.close();
}
```

Gestione

```
try {
   os = new ObjectOutputStream(fos);
   for (Complex complex: set.toArray()) {
      os.writeObject(complex.re());
      os.writeObject(complex.im());
   }
   return;
} catch (IOException ex) {
   log(fos,os,ex);
   return;
} finally {
   if (os != null) os.close(
   if (fos != null) fos.clos
}
```

Il codice è strutturato male, ma è bene notare che il blocco finally viene eseguito anche se il blocco try e catch terminano con la return

Problema

- Anche le close() lanciano eccezioni.
- Si potrebbe scegliere di loggare o meno l'evento eccezionale in chiusura.
- Ignorare una eccezione non è una buona pratica, se lo fate, almeno abbiate una giustificazione molto forte e scrivetela nel codice

Soluzioni

```
  } finally {
       try {
         if (os != null) os.close();
         if (fos != null) fos.close();
       } catch (IOException e) { /* ignored. */
/* in ogni caso giustificate perche' ignorate */ }
     }
```

```
  } finally {
       try {
         if (os != null) os.close();
         if (fos != null) fos.close();
       } catch (IOException e) { log(os,fos,e); }
     }
```

Nota

- Il metodo ha una struttura troppo complessa, meglio suddividerlo in altri metodi