```
}
```

Forma corretta:

```
void method (String filename){
      if (new File(filename).exists()){
            // Controlli di white list ….
            System.exec("more " + filename);
      }
}
```

7.2.10.8 Oggetti

Per ragioni di sicurezza è necessario rendere le classi e gli oggetti non clonabili. Di seguito viene riportato un esempio su come è possibile rispettare questa regola:

```
[...]
public final void clone() throws java.lang.CloneNotSupportedException {
  throw new java.lang.CloneNotSupportedException();
}
[...]
```

Nei casi eccezionali, che dovrebbero essere motivati e ampiamente documentati, bisogna etichettare i metodi che consentono la clonazione di tipo “final”, in modo da evitare potenziali un loro malevolo override. Di seguito viene riportato un esempio su come è possibile gestire queste eccezioni:

```
[...]
public final void clone() throws java.lang.CloneNotSupportedException {
  super.clone();
}
[...]
```

Comparazione degli oggetti di classe. Non effettuare mai la comparazione per nome degli oggetti di classe. Di seguito viene riportato un esempio su come è possibile rispettare questa regola:

Esempio:

Forma non corretta:

```
public class MyClass {
  public boolean sameClass (Object o) {
    Class thisClass = this.getClass();
    Class otherClass = o.getClass();
    return (thisClass.getName() == otherClass.getName());
  }
}
```

Forma corretta:

```
package esempio;
public class MyClass {
  public boolean sameClass (Object o) {
    Class thisClass = this.getClass();
    Class otherClass = o.getClass();
    return (thisClass == otherClass);
  }
}
```

7.2.10.9 Serializzazione e deserializzazione

Rendere le classi e gli oggetti non serializzabili. Di seguito viene riportato un esempio su come è possibile rispettare questa regola:

```
[...]
private final void writeObject(ObjectOutputStream out) throws java.io.IOException
{
   throw new java.io.IOException("L’oggetto non può essere serializzato ");
}
[...]
```

Rendere le classi e gli oggetti non deserializzabili. Di seguito un esempio su come è possibile rispettare questa regola:

```
[...]
private final void readObject(ObjectInputStream in) throws java.io.IOException {
   throw new java.io.IOException("L’oggetto non può essere deserializzato");
}
```