Logging

Prof. Michele Loreti

Programmazione Avanzata

Corso di Laurea in Informatica (L31)

Scuola di Scienze e Tecnologie

Logging

`Java` is equipped with a *logging system* that can be used to keep track of executions...

Logging

Java is equipped with a *logging system* that can be used to keep track of executions. . . and limit the us of System.out. println (...) !

Logging

Java is equipped with a *logging system* that can be used to keep track of executions. . . and limit the us of System.out. println (...) !

Logging system manages a default logger that we get by calling:

```
Logger.getGLobal()
```

Logging

Java is equipped with a *logging system* that can be used to keep track of executions. . . and limit the us of System.out. println (...) !

Logging system manages a default logger that we get by calling:

```
Logger.getGLobal()
```

A *logger* provides method to register relevant event of our application:

```
Logger.getGLobal().info("Opening file "+ filename);
```

Logging

Java is equipped with a *logging system* that can be used to keep track of executions. . . and limit the us of System.out. println (...) !

Logging system manages a default logger that we get by calling:

```
Logger.getGLobal()
```

A *logger* provides method to register relevant event of our application:

```
Logger.getGLobal().info("Opening file "+ filename);
```

The result is something of the form:

```
Apr 24, 2018 12:30:16 PM it.unicam.cs.pa.examples.
    ExExceptions data.txt
INFO: Opening file data.txt
```

Logging

In an application we can use different loggers that are associated with a name:

```
Logger logger = Logger.getLogger("com.mycompany.myapp");
```

Logging

In an application we can use different loggers that are associated with a name:

```
Logger logger = Logger.getLogger("com.mycompany.myapp");
```

The structure of the name recalls a hierarchy among the loggers.

Logging

In an application we can use different loggers that are associated with a name:

```
Logger logger = Logger.getLogger("com.mycompany.myapp");
```

The structure of the name recalls a hierarchy among the loggers.

Each logger is equipped with a level: OFF, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, ALL.

Logging

In an application we can use different loggers that are associated with a name:

```
Logger logger = Logger.getLogger("com.mycompany.myapp");
```

The structure of the name recalls a hierarchy among the loggers.

Each logger is equipped with a level: OFF, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, ALL.

You can log at the right level:

```
logger.log(level, message);
```

Logging

In an application we can use different loggers that are associated with a name:

```
Logger logger = Logger.getLogger("com.mycompany.myapp");
```

The structure of the name recalls a hierarchy among the loggers.

Each logger is equipped with a level: OFF, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, ALL.

You can log at the right level:

```
logger.log(level, message);
```

The level of displayed message can be set:

```
logger.setLevel(level);
```

Logger methods...

Logger methods...

log(...) Methods: take a log level, a message string, and optionally some parameters to the message string.

Logger methods...

log(...) Methods: take a log level, a message string, and optionally some parameters to the message string.

logp(...) Methods: are similar to the log methods, but also take an explicit source class name and method name.

Logger methods...

log(...) Methods: take a log level, a message string, and optionally some parameters to the message string.

logp(...) Methods: are similar to the log methods, but also take an explicit source class name and method name.

logrp(...) Methods: are similar to logp method, but also take an explicit bundle object to be used in localising the log message.

Logger methods...

log(...) Methods: take a log level, a message string, and optionally some parameters to the message string.

logp(...) Methods: are similar to the log methods, but also take an explicit source class name and method name.

logrp(...) Methods: are similar to logp method, but also take an explicit bundle object to be used in localising the log message.

Utility methods: for tracing method entries (the entering methods), method returns (the exiting methods) and throwing exceptions (the throwing methods).

Logger methods...

log(...) Methods: take a log level, a message string, and optionally some parameters to the message string.

logp(...) Methods: are similar to the log methods, but also take an explicit source class name and method name.

logrp(...) Methods: are similar to logp method, but also take an explicit bundle object to be used in localising the log message.

Utility methods: for tracing method entries (the entering methods), method returns (the exiting methods) and throwing exceptions (the throwing methods).

Log level methods: These methods are named after the standard Level names and take a single argument, a message string.

To be continued...