Best practices for code and architecture

The revamp clean old style code and old patterns which are now totally obsoletes for new style code and new pattern that will be obsolete in 2 years.

Coding rules

Iterator

Don't use iterators anymore

```
iter = collection.iterator;
while(iter.hasNext())
{
  element = iter.next();
}
```

Use the foreach form instead

```
for(Element element: collection)
{
}
```

Singletons

Don't use singletons anymore

```
ResourceManager.getInstance()
```

Instead, use *picoContainer* to create an object that will have only one instance.

For this, you simply need to declare the component in the container

```
pico = new PicoBuilder().withConstructorInjection().withCaching().build();
pico.addComponent(CompilerContainer.class, this);
...
pico.addComponent(ResourceManager.class);
```

When you need to get the instance, simply put the component as a constructor parameter

```
public MyClass(ResourceManager resourceManager, (other dependencies) ) {
  this.resourceManager = resourceManager;
}
```

Package and module design

Golden rule: no cyclic dependencies
Cyclic dependencies means that your class/package/module have bidirectionnal dependencies. Thus, you can't separate one element from the other, they are a real pain to maintenance, modularization and readability.

If you have a doubt on wether your classes respects this rule, you may use:

- the Dependencies Structure Matrix of Intellij (only available on ultimate version). It is an excellent tool to analyze dependencies and help to delete cycles.
- for Eclipse users, the ispace plugin seems to provide a way to analyze package dependencies.

Duplication

There were some duplications due to architecture limitations in the past.

Now, you can create a component that may execute the commons work and inject in several classes.

(example coming...)