

Coding Conventions

[TracNav](#)

- [JPFWiki](#) - Welcome Page

[Introduction...](#)

[Installing JPF...](#)

[User Guide...](#)

[Developer Guide](#)

- [Design](#)
- [Choice Generator](#)
- [Partial Order Reduction](#)
- [Attributes](#)
- [Listener](#)

[MJI...](#)

- [Bytecode Factory](#)
- [Logging](#)
- [Report](#)
- [Embedded](#)
- [JPF tests](#)
- [JPF project layout](#)
- [Create a JPF project](#)
- [Coding Conventions](#)
- [Hosting update site](#)

[Projects...](#)

- [Summer Projects](#)
- [External Projects](#)
- [Change\(B\)log](#)

[About...](#)

- [Events](#)
- [Presentations](#)
- [Papers](#)
- [FAQ](#)
- [History?](#)
- [Support](#)
- [People?](#)
- [Playground](#)
- [Table of Context](#)

JPF is an open system. In order to keep the source format reasonably consistent, we strive to keep the following minimal set of conventions

- 2 space indentation (no tabs)
- opening brackets in same line (class declaration, method declaration, control statements)
- no spaces after opening '(', or before closing ')'
- method declaration parameters indent on column
- all files start with copyright and license information
- all public class and method declarations have preceding Javadoc comments

- we use **camelCase** instead of ***underscore_names*** for identifiers
- type names are upper case

The following code snippet illustrates these rules.

```
/* <copyright notice goes here>
 * <license referral goes here>
 */

public class MyClass {

    /**
     * this is my public method example
     */
public void foo (int arg1, int arg2,
                  int arg3) {
    if (bar) {
        ..
    } else {
        ..
    }
    ..
}
```

We consider modularity to be of greater importance than source format. With its new configuration scheme, there is no need to introduce dependencies of core classes towards optional extensions anymore. If you add something that is optional, and does not seamlessly fit into an existing directory, keep it separate by adding new directories. The core JPF classes should not contain any additional dependencies to external code.