Design Patterns

1 - Factory

Path: net.sf.freecol/server/model/Session.java

A classe abstrata Session contém o método estático lookup, que permite criar instâncias de sessões com base em uma chave ou outros parâmetros. Por isso, é um padrão de factory.

```
/**
 * Look up a session of specified type given the game objects involved.
 *
 * @param <T> The actual session class found.
 * @param type The class of session.
 * @param o1 The first {@code FreeColGameObject} in the session.
 * @param o2 The second {@code FreeColGameObject} in the session.
 * @return A session of the specified type, or null if not found.
 */
 Mike Pope +1
public static <T extends Session> T lookup(Class<T> type,
    FreeColGameObject o1, FreeColGameObject o2) {
    return lookup(type, o1.getId(), o2.getId());
}
```

2 - Singleton

Path: net.sf.freecol/server/generator/River.java

Esse design pattern assegura que somente uma única instância de uma classe exista e disponibiliza um ponto centralizado para aceder essa instância. É normalmente caracterizado pela existência de um método estático que permite a obtenção da instância exclusiva.

```
public class River {

    16 usages
    private static final Logger logger = Logger.getLogger(River.class.getName());

    3 usages
    private final TileImprovementType riverType;

    /**
     * Possible direction changes for a river.
     * @see net.sf.freecol.common.model.Map
     */
```