Khassenov Aldiyar

SE-2413

Files in the repository (src/):

- **Main.java** is the entry point, examples of using the builder and World methods.
- **Minecraft.java** is the main class; fields are made private final; nested static Builder implements the Builder pattern; getters and toString().
- **World.java** is an auxiliary class that uses the builder to create objects.

What's done:

- Minecraft is a product class with private final fields: worldName, ip (String), mode, biome, playerName, modification.
- The nested static class MinecraftBuilder implements the fluent API: methods worldName(...), ip(...), mode(...), biome(...), playerName(...), modification(...) return this and accumulate the values, build() creates Minecraft.
- The World uses the builder to conveniently create objects (createWorld() and joinServer()).
- Main demonstrates the use of the builder and the World.

Clean Code principles applied:

- Clear names: all fields and methods in camelCase (worldName, playerName, etc.), which improves readability.
- Encapsulation: Minecraft fields are private final, externally accessible only through getters; the builder encapsulates the build process.
- Immutability: The product fields are declared final, which makes the object more predictable and safe to use.
- Fluent API: The builder returns this, which makes the object creation code easy to read and reduces the chance of errors in the order of parameters.
- Single Responsibility: Minecraft stores the state, MinecraftBuilder is responsible only for the build, World is responsible for the scripts for creating objects.