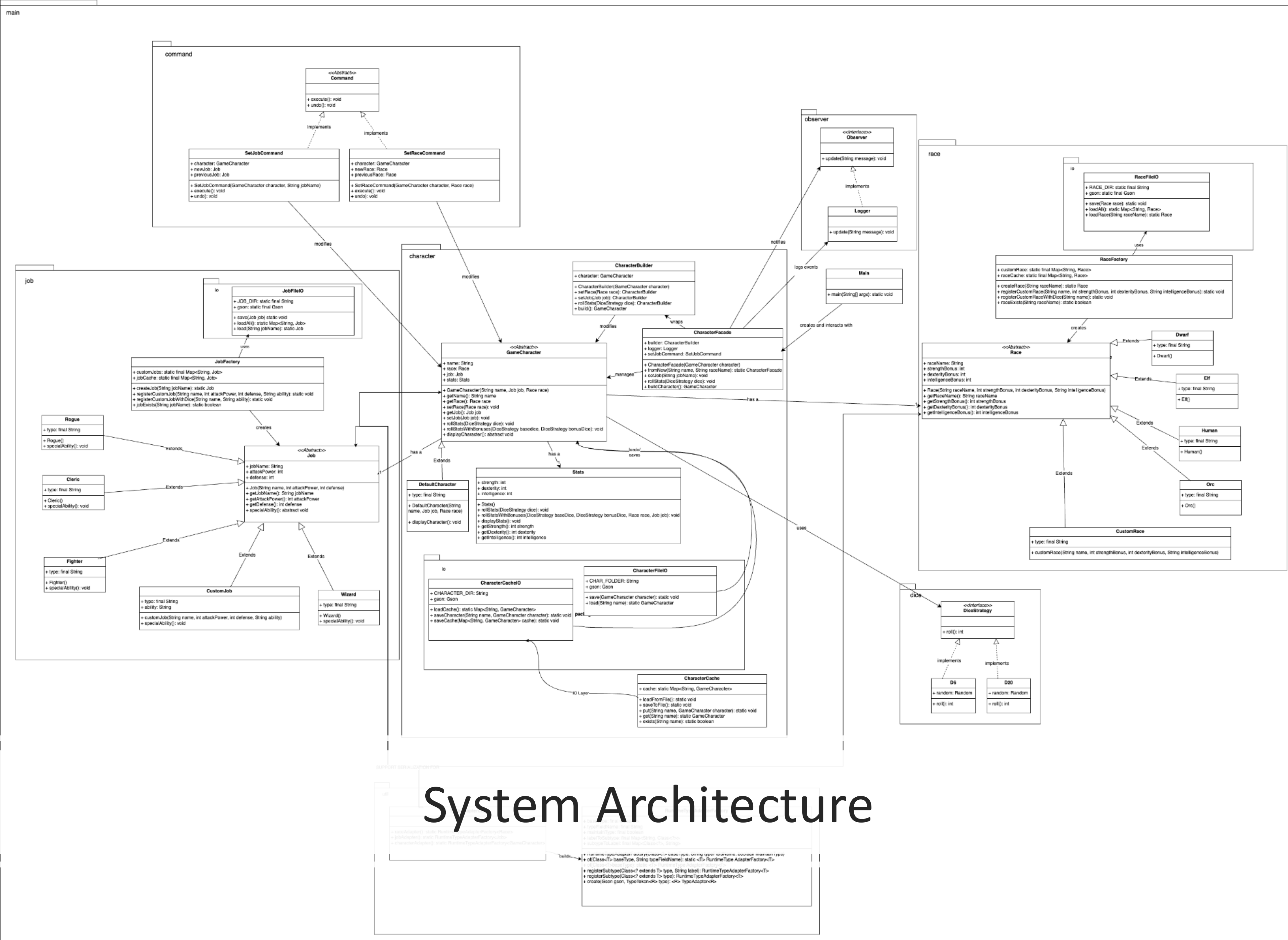
CS-665 Final Project

RPG Character Creation System designed with Software Design Patterns

Created by Natasya Liew U15913137
Spring 2025



main
command
<<Abstract>> Command
+ execute(): void
+ undo(): void
implements
implements
SetJobCommand
+ character: GameCharacter
+ newJob: Job
+ previousJob: Job
+ SetJobCommand(GameCharacter character, String jobName)
+ execute(): void
+ undo(): void
SetRaceCommand
+ character: GameCharacter
+ newRace: Race
+ previousRace: Race
+ SetRaceCommand(GameCharacter character, Race race)
+ execute(): void
+ undo(): void
modifies
modifies
observer
<<Interface>> Observer
+ update(String message): void
implements
Logger
+ update(String message): void
notifies
logs events
job
io
JobFileIO
+ JOB_DIR: static final String
+ gson: static final Gson
+ save(Job job) static void
+ loadAll(): static Map<String, Job>
+ load(String jobName): static Job
uses
JobFactory
+ customJobs: static final Map<String, Job>
+ jobCache: static final Map<String, Job>
+ createJob(String jobName): static Job
+ registerCustomJob(String name, int attackPower, int defense, String ability): static void
+ registerCustomJobWithDice(String name, String ability): static void
+ jobExists(String jobName): static boolean
creates
Rogue
+ type: final String
+ Rogue()
+ specialAbility(): void
Cleric
+ type: final String
+ Cleric()
+ specialAbility(): void
Fighter
+ type: final String
+ Fighter()
+ specialAbility(): void
CustomJob
+ type: final String
+ ability: String
+ customJob(String name, int attackPower, int defense, String ability)
+ specialAbility(): void
Wizard
+ type: final String
+ Wizard()
+ specialAbility(): void
Extends
<<Abstract>> Job
+ jobName: String
+ attackPower: int
+ defense: int
+ Job(String name, int attackPower, int defense)
+ getJobName(): String jobName
+ getAttackPower(): int attackPower
+ getDefense(): int defense
+ specialAbility(): abstract void
character
CharacterBuilder
+ character: GameCharacter
+ CharacterBuilder(GameCharacter character)
+ setRace(Race race): CharacterBuilder
+ setJob(Job job): CharacterBuilder
+ rollStats(DiceStrategy dice): CharacterBuilder
+ build(): GameCharacter
wraps
modifies
CharacterFacade
+ builder: CharacterBuilder
+ logger: Logger
+ setJobCommand: SetJobCommand
+ CharacterFacade(GameCharacter character)
+ fromNew(String name, String raceName): static CharacterFacade
+ setJob(String jobName): void
+ rollStats(DiceStrategy dice): void
+ buildCharacter(): GameCharacter
manages
Main
+ main(String[] args): static void
creates and interacts with
<<Abstract>> GameCharacter
+ name: String
+ race: Race
+ job: Job
+ stats: Stats
+ GameCharacter(String name, Job job, Race race)
+ getName(): String name
+ getRace(): Race race
+ setRace(Race race): void
+ getJob(): Job job
+ setJob(Job job): void
+ rollStats(DiceStrategy dice): void
+ rollStatsWithBonuses(DiceStrategy basedice, DiceStrategy bonusDice): void
+ displayCharacter(): abstract void
has a
has a
has a
Extends
loads/saves
uses
DefaultCharacter
+ type: final String
+ DefaultCharacter(String name, Job job, Race race)
+ displayCharacter(): void
Stats
+ strength: int
+ dexterity: int
+ intelligence: int
+ Stats()
+ rollStats(DiceStrategy dice): void
+ rollStatsWithBonuses(DiceStrategy baseDice, DiceStrategy bonusDice, Race race, Job job): void
+ displayStats(): void
+ getStrength(): int strength
+ getDexterity(): int dexterity
+ getIntelligence(): int intelligence
io
CharacterCacheIO
+ CHARACTER_DIR: String
+ gson: Gson
+ loadCache(): static Map<String, GameCharacter>
+ saveCharacter(String name, GameCharacter character): static void
+ saveCache(Map<String, GameCharacter> cache): static void
CharacterFileIO
+ CHAR_FOLDER: String
+ gson: Gson
+ save(GameCharacter character): static void
+ load(String name): static GameCharacter
IO Layer
CharacterCache
+ cache: static Map<String, GameCharacter>
+ loadFromFile(): static void
+ saveToFile(): static void
+ put(String name, GameCharacter character): static void
+ get(String name): static GameCharacter
+ exists(String name): static boolean
race
io
RaceFileIO
+ RACE_DIR: static final String
+ gson: static final Gson
+ save(Race race): static void
+ loadAll(): static Map<String, Race>
+ loadRace(String raceName): static Race
uses
RaceFactory
+ customRace: static final Map<String, Race>
+ raceCache: static final Map<String, Race>
+ createRace(String raceName): static Race
+ registerCustomRace(String name, int strengthBonus, int dexterityBonus, String intelligenceBonus): static void
+ registerCustomRaceWithDice(String name): static void
+ raceExists(String raceName): static boolean
creates
<<Abstract>> Race
+ raceName: String
+ strengthBonus: int
+ dexterityBonus: int
+ intelligenceBonus: int
+ Race(String raceName, int strengthBonus, int dexterityBonus, String intelligenceBonus)
+ getRaceName(): String raceName
+ getStrengthBonus(): int strengthBonus
+ getDexterityBonus(): int dexterityBonus
+ getIntelligenceBonus(): int intelligenceBonus
Dwarf
+ type: final String
+ Dwarf()
Elf
+ type: final String
+ Elf()
Human
+ type: final String
+ Human()
Orc
+ type: final String
+ Orc()
CustomRace
+ type: final String
+ customRace(String name, int strengthBonus, int dexterityBonus, String intelligenceBonus)
Extends
dice
<<Interface>> DiceStrategy
+ roll(): int
implements
implements
D6
+ random: Random
+ roll(): int
D20
+ random: Random
+ roll(): int
builds
+ of(Class<T> baseType, String typeFieldName): static <T> RuntimeTypeAdapterFactory<T>
+ registerSubtype(Class<? extends T> type, String label): RuntimeTypeAdapterFactory<T>
+ registerSubtype(Class<? extends T> type): RuntimeTypeAdapterFactory<T>
+ create(Gson gson, TypeToken<R> type): <R> TypeAdapter<R>

System Architecture

Design Patterns Used

- Factory Pattern (RaceFactory, JobFactory)
- Builder Pattern (CharacterBuilder)
- Facade Pattern (CharacterFacade)
- Strategy Pattern (D6, D20)
- Command Pattern (SetJobCommand, SetRaceCommand)
- Observer Pattern (Logger)
- Adapter Pattern (RuntimeTypeAdapterFactory)
- Cache Pattern (CharacterCache)

Summary

Pattern	Used In	Why It Was Needed
Factory	JobFactory, RaceFactory	To create built-in and custom jobs/races without modifying existing logic. Supports open/closed principle.
Builder	CharacterBuilder	To construct GameCharacter objects step-by-step with clear control over attributes.
Facade	CharacterFacade	To simplify character creation by exposing a unified interface and hiding complexity.
Strategy	DiceStrategy, D6, D20	To allow interchangeable dice logic for rolling stats. Promotes flexibility and reusability.
Command	SetJobCommand, SetRaceCommand	To encapsulate job/race changes and enable undo/redo functionality (extensible).
Observer	Logger	To decouple logging from character logic and track key updates in real time.
Adapter (Gson)	RuntimeTypeAdapterFactory + TypeAdapterUtil	To enable polymorphic serialization of abstract types (Race, Job, GameCharacter).
Cache	CharacterCache, CharacterCacheIO	To avoid reloading JSON files repeatedly and improve performance.

Factory Pattern Sample

```
public static Job createJob(String jobName) {
  switch (jobName.toLowerCase()) {
    case "fighter": return new Fighter();
    case "wizard": return new Wizard();
    ...
    default:
      return
customJobs.get(jobName.toLowerCase());
  }
}
```

- Centralizes object creation to **avoid hardcoding job instantiation** across the codebase.
- Makes it easy to **add new Job types** without modifying external logic — just extend the class and update the factory.
- Promotes **Open/Closed Principle**: code is open for extension, closed for modification.

Builder Pattern Sample

```
CharacterBuilder builder = new CharacterBuilder(character);
builder.setRace(race).setJob(job).rollStats(dice);
GameCharacter result = builder.build();
```

- Used to **gradually construct complex objects** (GameCharacter) step-by-step.
- Prevents constructor overloads by breaking down into logical, **chainable** actions.
- Helps maintain clean and flexible character creation workflows.

Façade Pattern Sample

```
CharacterFacade facade =
CharacterFacade.fromNew("hero", "elf");
facade.setJob("rogue");
facade.rollStats(new D20(), new D6());
GameCharacter character =
facade.buildCharacter();
```

- Hides internal complexity of multiple components (builder, job/race setting, stats) behind a **single interface**.
- **Simplifies usage** of multiple patterns for the end user or main application.
- Encourages **separation of concerns** and better encapsulation.

Strategy Pattern Sample

```
public interface DiceStrategy {
  int roll();
}

public class D20 implements DiceStrategy {
  public int roll() { return new
Random().nextInt(20) + 1; }
}
```

- Allows switching between **different dice-rolling algorithms** (e.g., D6 vs D20) at runtime.
- Enables **testing and tuning** of stat systems without changing underlying character code.
- Encourages modularity and reuse of logic.

Command Pattern Sample

```
Command setJob = new SetJobCommand(character, "wizard");
setJob.execute();
```

- Encapsulates job change actions as objects, supporting **undo/redo operations**.
- Useful for **history tracking**, debugging, and flexible game state management.
- Promotes **extensibility**, as new commands can be added without touching the core logic.

Observer Pattern Sample

```
Logger logger = new Logger();
logger.update("Job set to: wizard");
```

- Separates **logging/monitoring concerns** from business logic.
- Makes the system extensible for future observers (e.g., UI update, analytics).
- Keeps core logic clean and decoupled from side effects.

Polymorphic Serialization (Gson Type Adapter Factory)

```
Gson gson = new GsonBuilder()

.registerTypeAdapterFactory(TypeAdapterUtil.ch
aracterAdapter())
  .create();
```

- It **adapts the generic Gson deserializer** to handle **abstract or interface types** (Race, Job, GameCharacter).
- Without it, Gson cannot directly instantiate subclasses of abstract classes like Race or Job.
- Crucial for saving custom job/race and characters to the assets folder (supplement cache)

Caching Pattern Sample

```
CharacterCache.put("natasya", character);
GameCharacter cached =
CharacterCache.get("natasya");

CharacterCache.put("hero", character);
CharacterCache.get("hero");
```

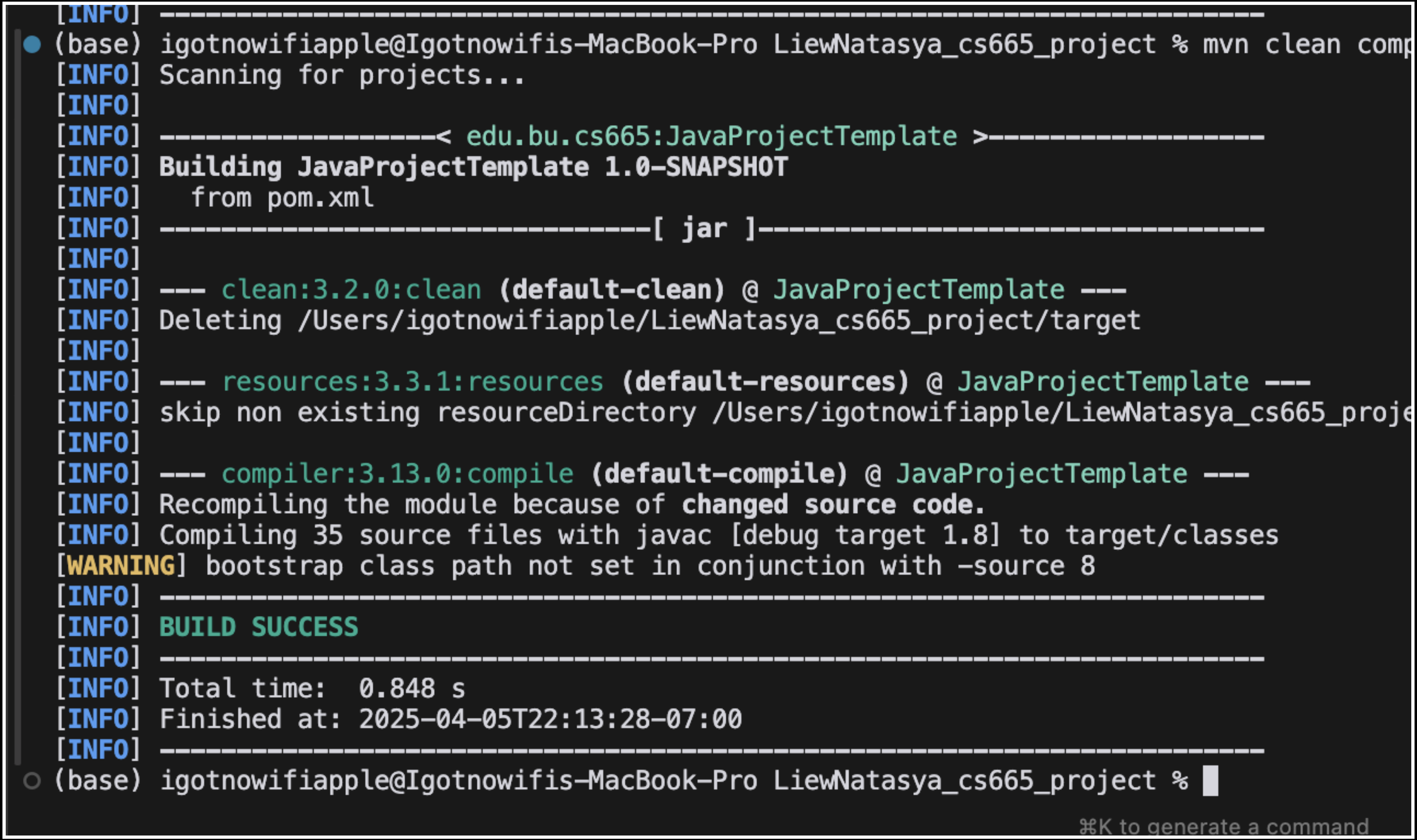
- Improves performance and **reduces file I/O** by reusing in-memory character data.
- Supports **quick lookups and preloading** without repeatedly deserializing files.
- Useful in games for fast access to previously used objects.

Best Practices & Quality

- JSON-based character persistence
- Modular and reusable codebase
- Javadoc-style documentation
- Defensive programming and exception handling
- Consistent checkstyle (Google Java)
- No SpotBugs issues

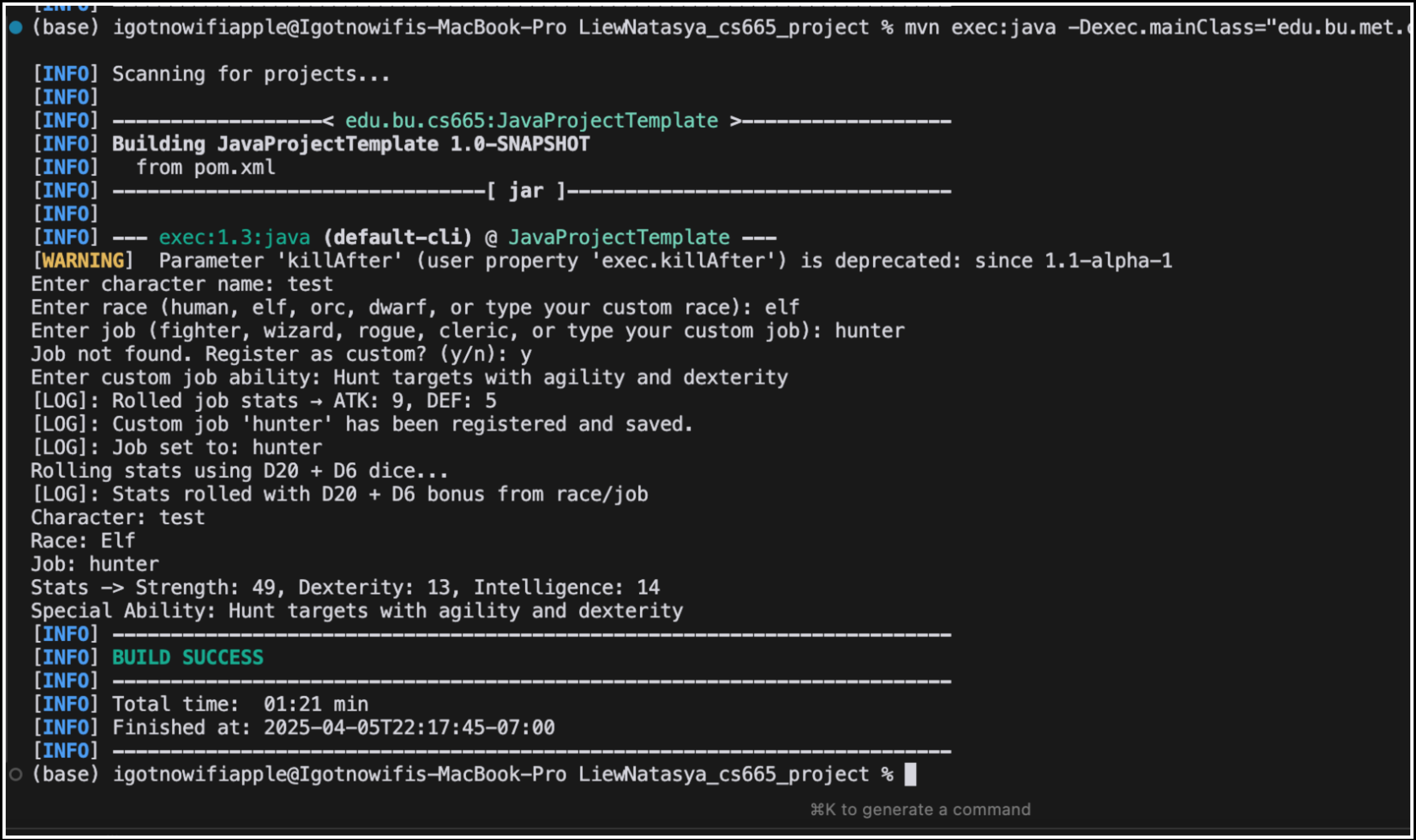
Key Learning Points

- **Design patterns don't always simplify implementation upfront**
  - Initially, patterns like **Facade**, **Command**, and **Factory** introduced more boilerplate code.
  - But they **significantly reduced coupling**, which made the code **much easier to fix and extend** when bugs appeared.
- **Adding features became more modular and intuitive**
  - New races and jobs could be added via JSON without touching core logic—thanks to **Factory Pattern** + **RuntimeTypeAdapterFactory**.
  - Supporting undo/redo or multiple dice configurations was just a matter of adding new classes implementing existing interfaces.
- **Refactoring was painful but extremely rewarding**
  - Refactoring across patterns (like consolidating dice rolls or serializing objects) introduced a risk of bugs.
  - But after refactoring, the system became **cleaner, scalable**, and much easier to reason about.
- **Pattern composition was powerful**
  - **Builder + Facade** helped manage the step-by-step creation of characters.
  - **Strategy + Factory** enabled dynamic behaviors and new logic without modifying existing implementations.
- **Debugging and testing became focused**
  - Patterns enabled **clear separation of concerns**, making it easier to write **unit tests per class**.
  - Bugs were often isolated to specific modules, reducing debugging scope.
- **Polymorphic serialization was non-trivial**
  - Implementing **RuntimeTypeAdapterFactory** required deep understanding of abstract types and Gson's limitations.
  - Once working, it enabled seamless save/load for any Job, Race, or Character.
- **Observer pattern encouraged good side-effect management**
  - Logging was cleanly abstracted, allowing you to observe important changes without touching business logic.
- **Overall: Patterns enhanced maintainability and scalability**
  - The final codebase was **ready for growth**, allowing other devs to add new classes or features without risk.
  - Design patterns made the system more **testable**, **clean**, and **future-proof**.



```
[INFO] ------------------------------------------------------------------------
● (base) igotnowifiapple@Igotnowifis-MacBook-Pro LiewNatasya_cs665_project % mvn clean comp
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< edu.bu.cs665:JavaProjectTemplate >------------------
[INFO] Building JavaProjectTemplate 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ JavaProjectTemplate ---
[INFO] Deleting /Users/igotnowifiapple/LiewNatasya_cs665_project/target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ JavaProjectTemplate ---
[INFO] skip non existing resourceDirectory /Users/igotnowifiapple/LiewNatasya_cs665_proje
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ JavaProjectTemplate ---
[INFO] Recompiling the module because of changed source code.
[INFO] Compiling 35 source files with javac [debug target 1.8] to target/classes
[WARNING] bootstrap class path not set in conjunction with -source 8
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.848 s
[INFO] Finished at: 2025-04-05T22:13:28-07:00
[INFO] ------------------------------------------------------------------------
○ (base) igotnowifiapple@Igotnowifis-MacBook-Pro LiewNatasya_cs665_project % 
                                                          ⌘K to generate a command
```

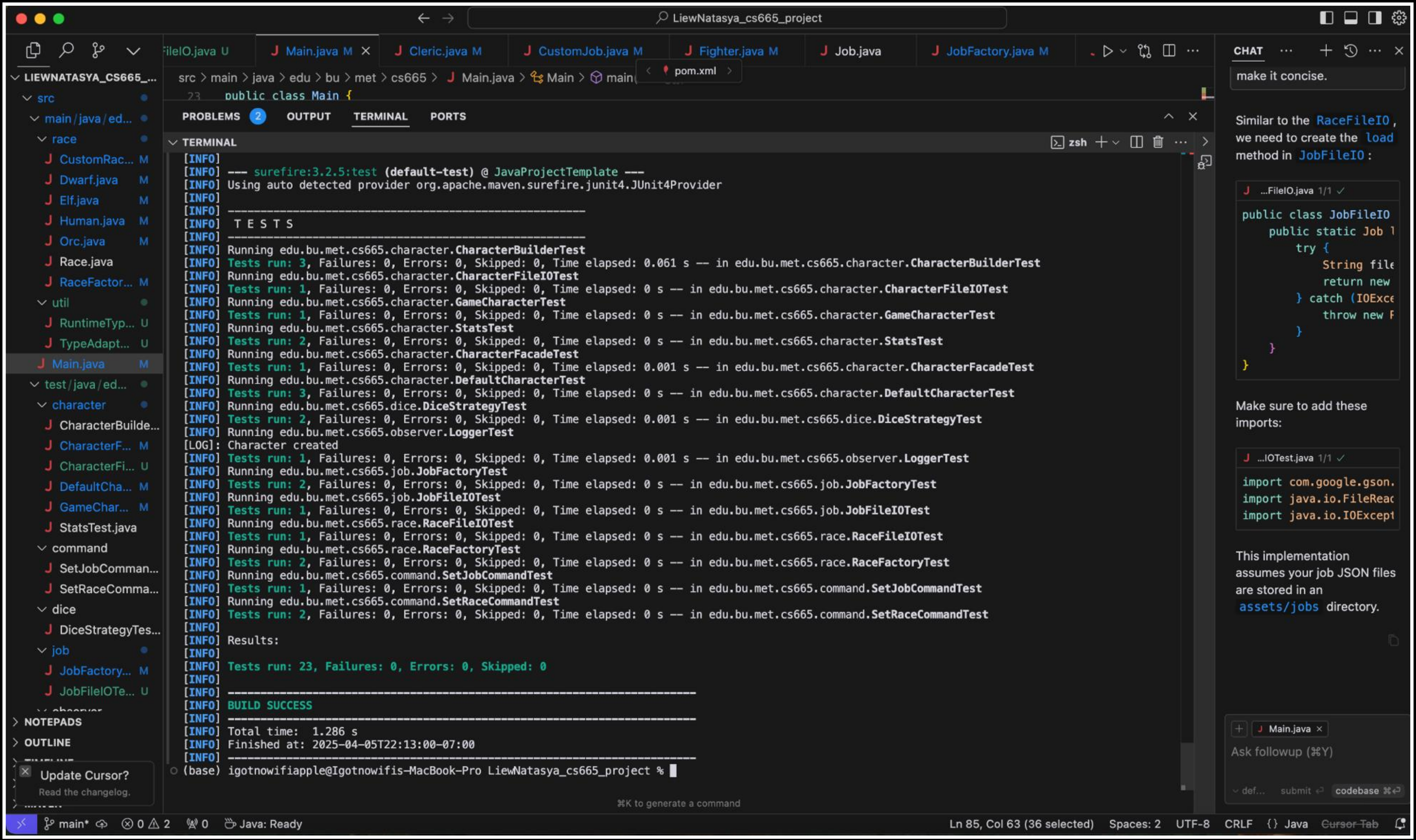
Compile & Build (Maven)



```
● (base) igotnowifiapple@Igotnowifis-MacBook-Pro LiewNatasya_cs665_project % mvn exec:java -Dexec.mainClass="edu.bu.met.

  [INFO] Scanning for projects...
  [INFO]
  [INFO] ------------------< edu.bu.cs665:JavaProjectTemplate >------------------
  [INFO] Building JavaProjectTemplate 1.0-SNAPSHOT
  [INFO]   from pom.xml
  [INFO] --------------------------------[ jar ]---------------------------------
  [INFO]
  [INFO] --- exec:1.3:java (default-cli) @ JavaProjectTemplate ---
  [WARNING]  Parameter 'killAfter' (user property 'exec.killAfter') is deprecated: since 1.1-alpha-1
  Enter character name: test
  Enter race (human, elf, orc, dwarf, or type your custom race): elf
  Enter job (fighter, wizard, rogue, cleric, or type your custom job): hunter
  Job not found. Register as custom? (y/n): y
  Enter custom job ability: Hunt targets with agility and dexterity
  [LOG]: Rolled job stats → ATK: 9, DEF: 5
  [LOG]: Custom job 'hunter' has been registered and saved.
  [LOG]: Job set to: hunter
  Rolling stats using D20 + D6 dice...
  [LOG]: Stats rolled with D20 + D6 bonus from race/job
  Character: test
  Race: Elf
  Job: hunter
  Stats -> Strength: 49, Dexterity: 13, Intelligence: 14
  Special Ability: Hunt targets with agility and dexterity
  [INFO] ------------------------------------------------------------------------
  [INFO] BUILD SUCCESS
  [INFO] ------------------------------------------------------------------------
  [INFO] Total time:  01:21 min
  [INFO] Finished at: 2025-04-05T22:17:45-07:00
  [INFO] ------------------------------------------------------------------------
○ (base) igotnowifiapple@Igotnowifis-MacBook-Pro LiewNatasya_cs665_project %
```

Run Output Demo



```
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ JavaProjectTemplate ---
[INFO] Using auto detected provider org.apache.maven.surefire.junit4.JUnit4Provider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running edu.bu.met.cs665.character.CharacterBuilderTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.061 s -- in edu.bu.met.cs665.character.CharacterBuilderTest
[INFO] Running edu.bu.met.cs665.character.CharacterFileIOTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s -- in edu.bu.met.cs665.character.CharacterFileIOTest
[INFO] Running edu.bu.met.cs665.character.GameCharacterTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s -- in edu.bu.met.cs665.character.GameCharacterTest
[INFO] Running edu.bu.met.cs665.character.StatsTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s -- in edu.bu.met.cs665.character.StatsTest
[INFO] Running edu.bu.met.cs665.character.CharacterFacadeTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s -- in edu.bu.met.cs665.character.CharacterFacadeTest
[INFO] Running edu.bu.met.cs665.character.DefaultCharacterTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s -- in edu.bu.met.cs665.character.DefaultCharacterTest
[INFO] Running edu.bu.met.cs665.dice.DiceStrategyTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s -- in edu.bu.met.cs665.dice.DiceStrategyTest
[INFO] Running edu.bu.met.cs665.observer.LoggerTest
[LOG]: Character created
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s -- in edu.bu.met.cs665.observer.LoggerTest
[INFO] Running edu.bu.met.cs665.job.JobFactoryTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s -- in edu.bu.met.cs665.job.JobFactoryTest
[INFO] Running edu.bu.met.cs665.job.JobFileIOTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s -- in edu.bu.met.cs665.job.JobFileIOTest
[INFO] Running edu.bu.met.cs665.race.RaceFileIOTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s -- in edu.bu.met.cs665.race.RaceFileIOTest
[INFO] Running edu.bu.met.cs665.race.RaceFactoryTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s -- in edu.bu.met.cs665.race.RaceFactoryTest
[INFO] Running edu.bu.met.cs665.command.SetJobCommandTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s -- in edu.bu.met.cs665.command.SetJobCommandTest
[INFO] Running edu.bu.met.cs665.command.SetRaceCommandTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s -- in edu.bu.met.cs665.command.SetRaceCommandTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 23, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.286 s
[INFO] Finished at: 2025-04-05T22:13:00-07:00
[INFO] ------------------------------------------------------------------------
(base) igotnowifiapple@Igotnowifis-MacBook-Pro LiewNatasya_cs665_project %
```

make it concise.

Similar to the `RaceFileIO`, we need to create the `load` method in `JobFileIO`:

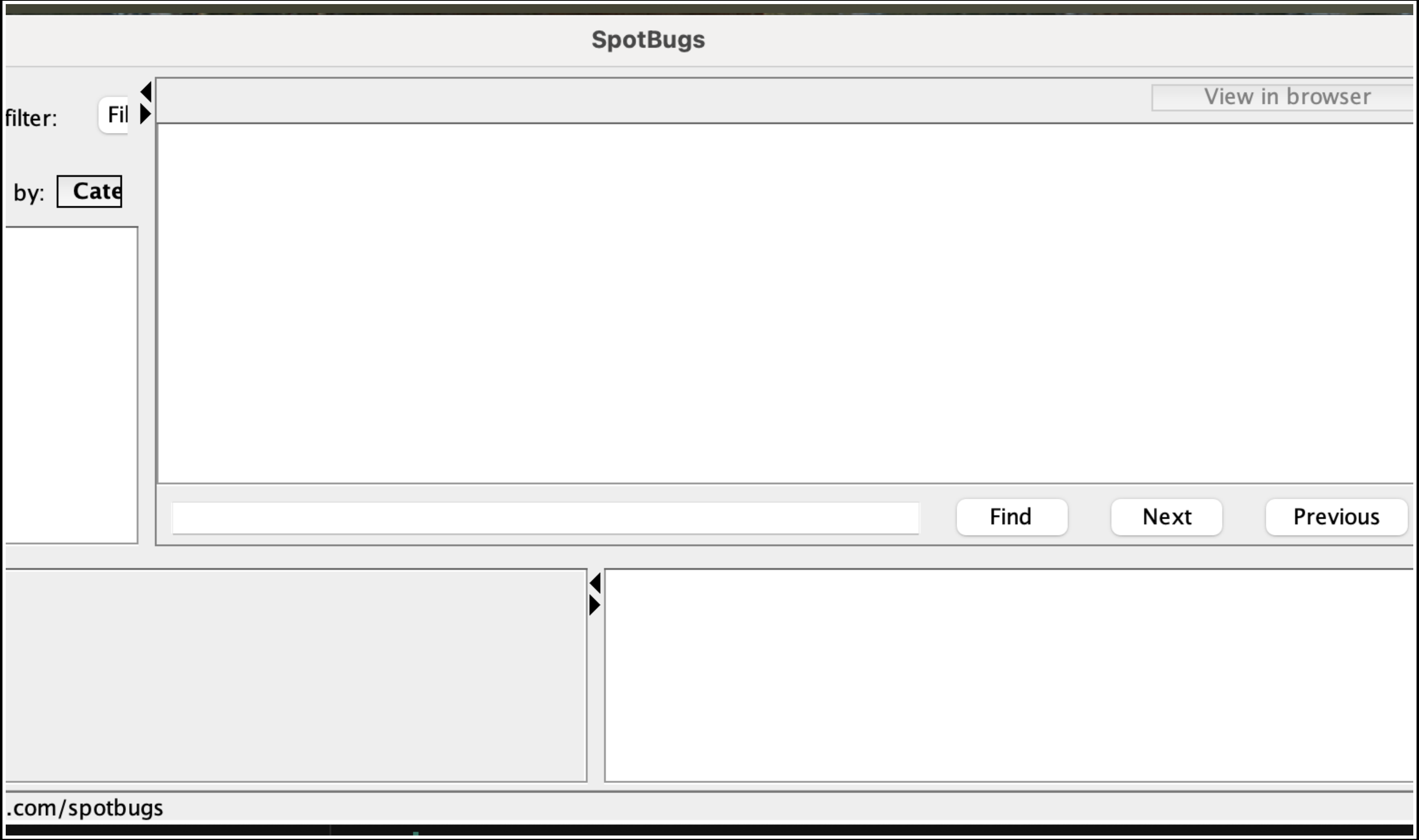
```
public class JobFileIO
    public static Job l
        try {
            String file
            return new
        } catch (IOExce
            throw new F
        }
    }
}
```

Make sure to add these imports:

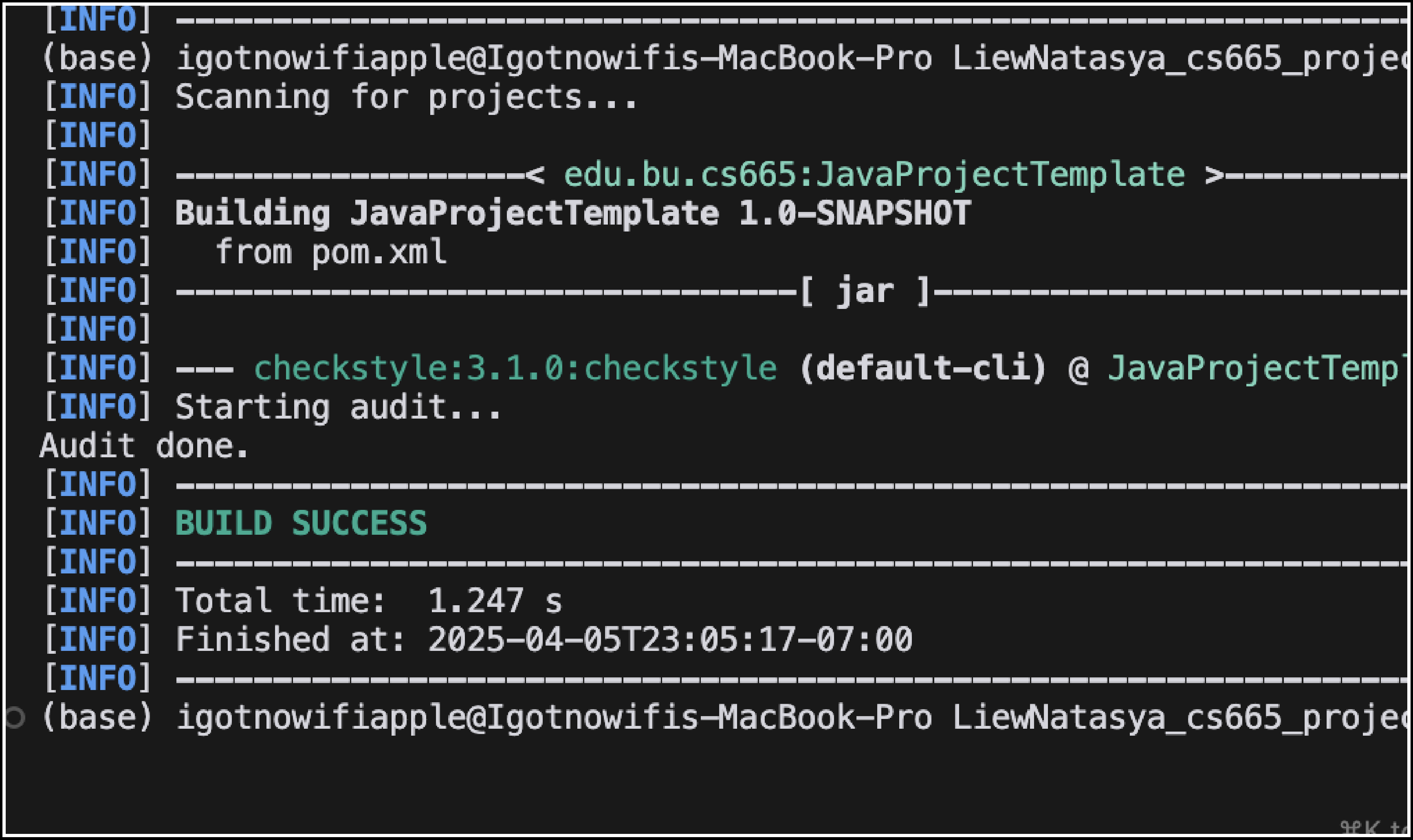
```
import com.google.gson.
import java.io.FileReac
import java.io.IOExcept
```

This implementation assumes your job JSON files are stored in an `assets/jobs` directory.

JUnit Testing



SpotBugs Scan



```
[INFO] ------------------------------------------------------------
(base) igotnowifiapple@Igotnowifis-MacBook-Pro LiewNatasya_cs665_proje
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< edu.bu.cs665:JavaProjectTemplate >---------
[INFO] Building JavaProjectTemplate 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]------------------------
[INFO]
[INFO] --- checkstyle:3.1.0:checkstyle (default-cli) @ JavaProjectTemp
[INFO] Starting audit...
Audit done.
[INFO] ------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
[INFO] Total time:  1.247 s
[INFO] Finished at: 2025-04-05T23:05:17-07:00
[INFO] ------------------------------------------------------------
(base) igotnowifiapple@Igotnowifis-MacBook-Pro LiewNatasya_cs665_proje
```

Checkstyle Validation