Development Environment

Last edited by **Priya Ashok Sardhara** 4 hours ago

Development Environment

Our team has carefully selected a comprehensive set of development tools, environments, and libraries to efficiently build the PawTopia virtual pet application. Our choices are based on compatibility, productivity, and adherence to the project requirements.

IDE and Development Tools Primary IDE: IntelliJ IDEA Ultimate 2023.3

• We've chosen IntelliJ IDEA as our primary integrated development environment for its robust Java development capabilities, excellent code refactoring tools, and built-in version control integration.

• All team members will use the same version to ensure consistency in project configuration and development experience.

• IntelliJ's GUI Designer will be utilized to help create and manage our Java Swing interface components, though all generated code will be properly documented and cited.

Version Control System: GitLab

• As required by the project specifications, we'll use the Western University GitLab instance for code repository management.

• We'll utilize GitLab's issue tracking system to manage tasks, track bugs, and document project progress.

• We'll maintain a detailed wiki documenting our design decisions, technical specifications, and project guidelines.

Build Tool: Apache Maven 3.9.5

• Maven will be used for dependency management, build automation, and project structuring.

• This ensures consistent builds across all team members' environments and simplifies the addition of external libraries.

• Project will be organized according to the standard Maven project structure.

Core Java Technologies

Java Development Kit (JDK): JDK 23

• In accordance with requirement 3.2.1, we'll use JDK 23 (the current version as of project start).

• All code will be written to be compatible with this version and will run on Windows 11 systems with standard Java installations.

User Interface Framework: Java Swing

• We've selected Java Swing for our GUI implementation due to its built-in nature in the Java standard library, reducing external dependencies.

• While Swing is more mature and less modern than JavaFX, it provides sufficient capabilities for our requirements without adding complex dependencies.

• This choice aligns with recommendation 5.4 in the project specification.

Testing Framework: JUnit 5

• As specified in requirement 3.2.11, we'll use JUnit 5 for unit testing our application.

• Test cases will be developed for all non-UI components of the application.

• We'll aim for high test coverage of the business logic layer.

Documentation: Javadoc

• Following requirement 3.2.10, all code will be extensively documented using Javadoc.

• Every class will include detailed class-level documentation.

• All public methods will have comprehensive Javadoc comments including parameter descriptions, return values, and exception details.

External Libraries

JSON Processing: Google Gson 2.10.1

• For handling JSON data serialization and deserialization as outlined in our File Format section.

• Gson provides straightforward and intuitive API for working with JSON in Java.

• This will enable us to efficiently implement the save/load game state functionality (requirement 3.1.5).

Graphics and Image Processing: Java Advanced Imaging API (JAI)

• For handling sprite manipulation and basic animation effects.

• Will assist with implementing the pet sprite animations described in requirement 3.1.10.

Time Management: Java Time API

• Built into the JDK, we'll use this for implementing cooldown timers for commands and tracking play session duration.

• Important for implementing the parental controls feature (requirement 3.1.11).

Logging: SLF4J with Logback

• For application logging and debugging support.

• Will help in troubleshooting during development and potentially provide diagnostic information for users.

Development Practices

Coding Standards

• We've established a team-specific coding standard based on the Google Java Style Guide.

• Consistent naming conventions, indentation, and code organization will be maintained across all source files.

• Code reviews will be conducted through GitLab merge requests to ensure adherence to these standards.

Documentation Strategy

• In addition to Javadoc, we'll maintain comprehensive documentation in the GitLab wiki.

• Design decisions, meeting minutes, and implementation details will be documented.

• User documentation will be created and included within the application.

Testing Strategy

• Unit tests will be written for all model classes and business logic.

• Integration tests will verify correct interactions between components.

• Manual UI testing will be conducted according to a defined test plan.

Deployment and Execution

Packaging: JAR with Dependencies

• The final application will be packaged as an executable JAR file that includes all dependencies.

• This will ensure that the application is self-contained and easy to run on any Windows system with Java installed.

Installation Requirements

• The application will require only a standard Java 23 (or newer) installation on Windows 11.

• No additional system configurations or installations will be needed.

• The application will be designed to be portable and not create files outside its installation directory.

Overall

This development environment has been carefully selected to meet all the requirements specified in section 3.2 of the project specification while providing the team with efficient tools for successful implementation of the PawTopia virtual pet application.