**Final Capstone Project: Drools Rule Engine Application**

**Objective:**

This lab exercise aims to synthesize the knowledge and skills you've acquired throughout the Drools course. Working in pairs, you will conceptualize, design, and implement a real-world application using the Drools rule engine, demonstrating your understanding of rule-based systems and their application in solving complex business problems.

**Team Formation:**

* **Students should work in pairs.**
* **Pairing Criteria:** All students are developers. Non-Java developers must pair with a Java developer in the group to ensure that all technical aspects of the project are adequately addressed.

**Part 1: Conceptualization**

* **Task:** Together with your partner, brainstorm and select a business concept where the Drools rule engine can be applied effectively. Consider areas such as finance, healthcare, e-commerce, insurance, or any domain with complex decision-making processes.
* **Output:** A brief document (1-2 pages) describing your chosen business concept, why it's suitable for a rule engine, and how Drools can improve or solve the business problem.

**Part 2: Domain Modeling**

* **Task:** Plan the domain objects and data structures required for your project. You must define a minimum of three domain objects. You may also create additional fact objects as necessary for your business logic.
* **Output:** A document outlining your domain model, including UML class diagrams or similar representations for each domain object and fact object (if used). Also, describe the relationships between these objects.

**Part 3: Rule and Session Planning**

* **Task:** Design the rules and sessions for your application. Your rules should:
  + Include cross-product evaluation to demonstrate complex decision-making between different domain objects.
  + Utilize execution control techniques such as salience, agenda groups, or activation groups to ensure rules fire in the correct order.
  + Implement derived facts to infer new information from existing data points.
  + Follow the best practices covered in the course regarding rule organization, reusability, and maintenance.
* **Requirement:** Your application must use at least one stateful and one stateless session to demonstrate your understanding of their differences and use cases.
* **Output:** A detailed plan of your rules and sessions, including pseudo-code or detailed descriptions of each rule's logic, the conditions under which they fire, and the expected outcomes. Also, specify which session(s) each rule will be part of and why.

**Part 4: Implementation and Execution**

* **Task:** Based on your plans in Parts 2 and 3, create the Java classes for your domain objects, write the Drools rules, define the KIE sessions, and integrate everything into a running application.
* **Requirements:**
  + Code quality and organization will be evaluated. Ensure your code is well-commented, and classes and rules are logically organized.
  + Your application must compile without errors and execute the rules as expected.
* **Output:** A fully functional Drools application that:
  + Implements the business concept chosen in Part 1.
  + Utilizes the domain model designed in Part 2.
  + Executes the rules and sessions planned in Part 3, demonstrating successful rule firing and the correct application of stateful and stateless sessions.
* **Demonstration:** Present your application to the class, showing proof of successful rule firing. Explain your decision-making process, the challenges faced, and how you overcame them. Your presentation should include a live demonstration of the application and a code walkthrough.

**Evaluation Criteria:**

* Creativity and relevance of the business concept.
* Complexity and correctness of the domain model.
* Logical and efficient rule and session design.
* Code quality, readability, and organization.
* Successful execution of the application and rule firing.

This capstone project is your opportunity to showcase your Drools expertise. Approach it with creativity and attention to detail. Good luck!