A) Clean code development

1. The Destination class defines fields like country, city, weather, budget, and activities, which are intuitive and self-explanatory. These names clearly represent the purpose of each field, reducing the need for additional comments or explanations.

2. In the Destination class, the fields are private, and getters/setters are used to access them. This ensures that the internal state of the class is protected, and any changes are controlled through the setter methods, adhering to object-oriented principles.

3. Both NextDestination.java and Destination.java files follow consistent indentation and spacing conventions. Consistent formatting improves readability, making it easier for developers to navigate and understand the codebase.

4. The Destination class is solely responsible for holding destination data, while logic for filtering and processing is handled separately in NextDestination.java.

5. The destinations are loaded dynamically from a JSON file (destinations.json) rather than hardcoding data in the program. This improves scalability and makes it easier to update data without modifying the source code.

B) Personal CCD cheat sheet

1. Names should clearly indicate their purpose.
2. Stick to consistent naming conventions.
3. Keep functions small and focused. Ideally, they should perform one task.
4. Use private fields with public getters and setters to control access.
5. Clean up unused variables, imports, or methods.
6. Break down complex logic into simpler, named methods or use helper functions.
7. Implement exception handling to ensure the app behaves predictably during runtime errors.
8. Write unit tests to verify the behavior of individual components.
9. Avoid deeply nested conditions. Use early returns or split logic into smaller functions.
10. Avoid duplicating code. Refactor shared logic into reusable methods.
11. Group related methods and classes logically.
12. Avoid unnecessary loops or difficult operations inside frequently called methods.
13. Leverage built-in libraries and frameworks instead of building solutions on your own.