UML Class Diagrams

Please draw a UML Class diagram for each of the following stories:

Story 1:

Once upon a time, there was a little girl named Lily who loved to play with her toys. Her favorite toy was a stuffed bear named Teddy. Lily and Teddy were inseparable and did everything together.

- Step 1: Lily drew a UML diagram to represent the relationship between herself and Teddy. She included an association relationship to show that she was associated with Teddy.
- Step 2: One day, Lily went to a toy store and found a toy car that she really liked. She bought the car and started playing with it along with Teddy. She drew a UML diagram to represent the relationship between herself, Teddy, and the toy car. She used an aggregation relationship to show that the toy car was part of a collection of toys that included Teddy.
- Step 3: As time passed, Lily's collection of toys grew. She acquired more stuffed animals, toy cars, and dolls. She drew a UML diagram to represent the relationship between herself and her toys. She used a composition relationship to show that she owned the toys and that they couldn't exist without her.
- Step 4: One day, Lily's parents decided to give her a toy chest to store all of her toys. Lily drew a UML diagram to represent the relationship between herself, the toy chest, and her toys. She used a dependency relationship to show that the toy chest depended on her to store her toys.
- Step 5: Over time, Lily grew older and lost interest in her toys. She donated them to a children's hospital so that other children could enjoy them. She drew a UML diagram to represent the relationship between herself, the hospital, and her toys. She used an association relationship to show that the hospital was associated with her toys.

Story 2:

Once upon a time, there was a student named John who enrolled in a university to pursue a degree in computer science. John was excited to start his classes and meet new people.

- Step 1: John drew a UML diagram to represent the relationship between himself and the university. He included an association relationship to show that he was associated with the university.
- Step 2: On the first day of classes, John was assigned to a computer science course taught by a professor named Dr. Smith. He drew a UML diagram to represent the relationship between himself, the university, Dr. Smith, and the course. He used an aggregation relationship to show that the course was part of a collection of courses offered by the university.
- Step 3: As the semester progressed, John attended classes in different classrooms across campus. He drew a UML diagram to represent the relationship between himself, the university, the courses, and the classrooms. He used a composition relationship to show that the classrooms were owned by the university and that they couldn't exist without it.

- Step 4: At the end of the semester, John received grades for his coursework. He drew a UML diagram to represent the relationship between himself, the university, the courses, and the grades. He used a dependency relationship to show that the grades depended on the university's grading system and on his performance in the courses.

- Step 5: The following semester, John registered for a new set of courses. He drew a UML diagram to represent the relationship between himself, the university, and the courses. He used an association relationship to show that he was associated with the courses he registered for.

- Step 6: In one of his courses, John worked on a group project with other students. He drew a UML diagram to represent the relationship between himself, the university, the course, the professor, the classroom, and the other students. He used various relationship types to show how all these objects were related.

Story 3:

A university management system needs to be developed to manage student enrollment, course offerings, grading, and registration. The system should have the following requirements:

1. The system should allow students to enroll in courses offered by the university.

- The system should have a Student class.
- The Student class should be associated with the university.
- The university should have a Course class.
- The Student class should have an aggregation relationship with the Course class to show that the courses are part of a collection of courses offered by the university.

2. The system should be able to assign students to courses taught by professors.

- The system should have a Professor class.
- The Course class should have a professor assigned to it.
- The Student class should have an association relationship with the Professor class to show that the student is associated with the professor.

3. The system should be able to assign classrooms to courses.

- The system should have a Classroom class.
- The Course class should have a classroom assigned to it.
- The Classroom class should have a composition relationship with the university to show that it is owned by the university.

4. The system should be able to store and calculate grades for each student.

- The system should have a Grade class.
- The Student class should have a collection of Grade objects to store the student's grades for each course.
- The Grade class should have a dependency relationship with the university's grading system.

5. The system should allow students to register for courses.

- The system should have a Registration class.

- The Registration class should have an association relationship with the Student class and the Course class to show that the student is associated with the courses they registered for.

6. The system should be able to manage group projects for courses.

- The system should have a Group class.
- The Course class should have a collection of Group objects to manage group projects for the course.
- The Group class should have association relationships with the Student class, the Professor class, the Course class, and the Classroom class to show how all these objects are related.