# Project ZOO

This year we'll do one project in two (separately graded) parts.

## General description

The theme of this project will be to provide an inventory system for the (non-human) animals of the NC Zoo (nczoo.org). We are only interested in the animals that are (or might be) on exhibit. We certainly can't actually store every animal, so in the course of developing your program, provide a system that could be extended to store everything, even though it will have only a token number of actual animal records.

We also can’t fully design or implement everything we’d like. Be severe, and state what you’ll try to accomplish. Organize your work so that if you can’t accomplish everything you’re trying to do, you have a way to omit some tasks and still have a useful product. Also list things that you aren’t going to attempt, but know how you’d approach them. For example, “we decided we didn’t have time to include an animal’s color, but if we could, we’d put that information in class *Abcd* and create a method in class *Efgh* to display it, and class *Ijkl* to sort by it.

## Student Groups

You will be placed in groups of 3 or 4. As a group you will decide the roles for each group member.

For part 1, the roles will be

1. Infrastructure Designer will be responsible for designing the actual data store for animal information. The infrastructure is responsible for persisting the data so that it can be reloaded at subsequent program startups. The infrastructure must also be flexible enough that new animal types can be accommodated with no program changes. (If the infrastructure can handle lions and tigers in the Africa exhibit, for example, adding bobcats to the North America exhibit should require no changes to the infrastructure.)
2. Animal Record Designer will be responsible for designing the actual individual animal information. As mentioned earlier, only a token number of animals will be represented, but these samples should vary widely.
3. User Interface Designer will specify what capabilities a user of this program will have. Regardless of whether the UI is text based or a GUI, users of this program will want certain capabilities. Creating animal records, listing animal records, sorting animal records by some criteria, and displaying subsets of animal records are some examples of these capabilities. The UI will have to interact with the infrastructure to accomplish these tasks. For example, asking "What mammals do you have in North America?" Or "What cats do you have in any section?" Or "Sort your birds by weight." The types of questions you support will have to be limited, and carefully decided upon.
4. Quality Assurance, if there is a fourth team member, will design the testing. This includes more than just JUnit tests and special purpose main methods to verify that the UI can create a bear, that there's an animal record that can represent a bear, and that the infrastructure can store, persist and restore a bear. It means creating use cases (like the bear use case above) to maximize test coverage of the program's function.

## Your Deliverables for Part 1

The big deliverable for Part 1 is an **Interface and Design document**. No code will need to be produced, but you will need to document

1. UI capabilities for the user, the major UI classes and interfaces, how the user will navigate the UI.
2. How the UI will produce and consume and manipulate Animal Designs
3. How the UI will manipulate the infrastructure
4. Animal Design classes and interfaces
5. Listing the animals that will be supported
6. Infrastructure classes and interfaces that will support the Animal Designs
7. Infrastructure design for storing data, and the techniques for persisting the data
8. Test use cases and strategy for as much software testing as possible.

As with our earlier assignments that involved your designing software, the ultimate goal of this document is that someone else, using only this document, could implement the system you envision.

Class and interface documentation may take the form of Unit 03’s PortfolioHolding Subclass description, if you wish. The UI should describe user capabilities and give use cases. Testing should also be expressed in use cases.

Note that each team member is committing to living with the team’s design. Just as an example, if the infrastructure isn’t providing the UI with sufficient capabilities to do the UI’s function, the team as a whole and individually has a problem.

### Grading the Design

Here are some items I will look for in grading your design:

1. Are the use cases representative of things a user will want to do? Individual use cases should be simple, but as a collection they should touch on all the major functions a user will need.
2. Does the UI design support all of the use cases’ requirements?
3. Is the UI program constructed so that its various operations can be combined in various ways to perform the necessary user operations? Are examples provided?
4. Does the UI specify what data is given to other program components, how that data is represented, and how the UI and the other components communicate? I.e., what interfaces does the UI provide to other components, and what interfaces must other components supply for the UI?
5. Is the animal representation structure extendable to include other types of animals, other locations of animals, etc?
6. Is there an extendable design for sorting and subsetting the animal inventory?
7. Does the infrastructure have clear, sufficient and extendable interfaces and operations to support the UI and animal representations?
8. Are there concrete examples in which some use cases are traced from the goal of the user, to the user’s interactions with the UI, to the UI’s use of specific animal representations and infrastructure capabilities, and to the infrastructure’s actions to accomplish the user’s intent? These examples should be concrete, listing classes, fields and methods.
9. Is there a test plan for each major component of the system? What tools will be used? What will be the specific test cases?

Be specific about what you will include, and what you will exclude. Short notes on what parts of the design can be easily extended for capabilities not included in the design are good.

Don’t give in to mission creep. Producing a small diamond is better than winding up with a dump truck of coal, even if the dump truck contains more carbon atoms.

Twice weekly updates from each team member

Twice week each team member will post a one or two sentence answer to each of these three questions:

1. What am I doing right now?
2. What will I do next?
3. What roadblocks are preventing me from making progress?

The deadlines for these updates will be decided by each team. Answers should be short, but specific.

### End of the project report

At the end of the project you'll submit a confidential report on each group member, including yourself: You should address the challenges the member faced, what they did or did not do to address those challenges, what they accomplished, what difficulties they left behind--either in the project or for others to pick up. Note that your individual grade will, in part, depend on this critique. If you report that "so and so was great," but his or her part of the project is mediocre, you will be held responsible for part of that mediocrity.

## Part 2

Part 2 will involve implementing your design using the interfaces you documented. More on that later.

## Meetings and Other Tools

Webex.com offers a free 2 week trial of their Premium service, which I’ve used and it works superbly. After the trial is over, another team member could sign up for a trial.

GoToMeeting.com offers their 30 day trial of their Pro plan. I’ve not used GoToMeeting.

GitHub.com offers free source code control, but don’t recommend that solution unless someone is already familiar with it. Easier alternatives are cloud storage (DropBox, Google Drive, or whatever Microsoft’s offering is), or to use zip and attach archive files to your forum posts. Blackboard can sometimes corrupt files named “.java”, so always zip them for protection.