IT 145 Introduction to Specification Documents

What Is a Specification Document?
As you read through the projects for this course, you will notice that each project has a specification document. Specification documents, formally referred to as software requirements specification (SRS) documents, provide a description of software **functionality**: what a piece of software should do.

Who Creates a Specification Document?
In the workforce, these documents might be created by the client requesting the software, the software design team, or a combination of the two. When specification documents are generated by a client, they can come in many forms. There might be outlines, tables, narratives, pictures, or some combination of those components. Sometimes they can be ambiguous and may require further meetings with your client. When the document is generated by a software engineer or other professional entity, it will contain nearly every detail needed to create the software: diagrams, database requirements, class structures, data types, logic, and so on.

In this course, you will not need to design your own specification documents, but will instead be reading and interpreting them. The specification documents for your projects are streamlined, focusing only on the information you will need for the tasks you have been assigned.

How Would a Developer Use a Specification Document?
Once created, a specification document would be given to the software development team to review, informing the software's design and architectural pattern. Each specification document looks a little different. When reviewing the specification documents for the projects in this course, there are a few key pieces that you want to look for.

- **Background information** tells you who the company is and what it does. This information gives you context so that you understand how the software you develop fits into the work of the company.

- **Software requirements** include descriptions of components (physical things such as pets, animals, or ships) your program should contain, as well as descriptions of functionality—what your program should *do*.

Interpretation Walkthrough
Let's discuss a section of the Pet BAG Specification Document from Project One, which describes the client's desired functionality. Here are some questions you should keep in mind as you read:

- What process is being described here? What are the steps of this process?
- What things are being used in this process?

Pet Check In

To update our current processes, we outline our current manual check-in process below, which involves multiple steps and verifications.

First, we determine whether the pet is a dog or a cat. Next, we determine if boarding space is available for the pet. There are currently 30 spaces for dogs and 12 spaces for cats. We would like the ability to update these settings as needed. If boarding space is available, we identify whether the pet is a new or returning visitor. If the pet has stayed with us before, we are seeking the ability to update information as needed. If the pet is a new visitor, we would like the opportunity to collect all the appropriate information.

Additionally, we would like the ability to gather information on the length of stay for each pet and if grooming services are required. Grooming services are only offered for dogs that stay at Pet BAG for two or more days. No grooming services are offered for cats.

Our final step is to assign the pet to a space.

In just the first few sentences, there is some key information about what classes (categories of objects) your program will use, how they are related, and how you will be designing a process that involves these classes. As you read through the section, you might have noticed the following:

- The section is titled "Pet Check In," and that sounds like a process. Processes are turned into **methods.**

- The start of the second paragraph contains three key nouns: pet, dog, and cat. Based on the sentence structure, we know that dogs and cats are both kinds of pets. That is a key indicator that Pet will be a **parent class** to the Dog and Cat **classes**.

The rest of the section lays out what the pet check in process looks like. This section could be used to help you plan out your code; it gives you the information you need to design pseudocode or a flowchart.

Finally, the table below lists some additional things you might have noticed from the example above. You will touch on these concepts (attributes, methods, conditional statements, and so on) more as you progress through the course.

Words to look for. . .	**How a software developer interprets them. . .**
"Gather information on. . ."	This is followed by specific data elements, which will become **attributes** in the appropriate class.
"Assign the pet to a space."	"Assign" is a verb, which usually indicates that a **method** is needed.
"If boarding space is available . . ."	Usually, when there is an "if" in the narrative, there will be a **conditional statement** in the code. How many "if" statements can you find?
"30 spaces . . . 12 spaces"	These are **constraints** and would require variable declaration and assignment.
"[our process involves] multiple steps and **verifications.**"	The word "verifications" implies that there will be some sort of **data validation** or **logic check.**