Abstraction and Hierarchy

What is Abstraction?

- Abstraction is one of the essential parts of Computer Science. It includes filtering out characteristics that we do not need so that we can concentrate on those that we do.
- It is also the filtering out of specific details.
- After filtering out unnecessary details and characteristic, we can create a model of what we are trying to solve.

Example of Irrelevant Characteristics

Below is a list of the characteristics of a cat. We want to remove unneeded **specific characteristics** to build a more universal model that can be used to categorize all cats. We can see how we do not need the specific characteristics to still be able to determine that it is a cat.

Specific Characteristics

- Brown Fur
- Long Fur
- Four Legs
- Tail
- Ears
- Two Eyes
- Likes Fish
- Meows
- Whiskers
- Fur

Remove Specific Characteristics

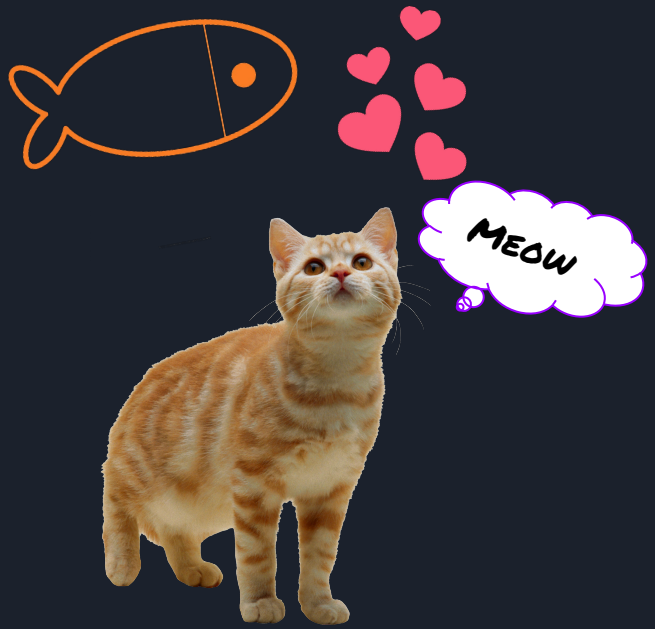
- ~~Brown Fur~~
- ~~Long Fur~~
- Four Legs
- Tail
- ~~Ears~~
- Two Eyes
- ~~Likes Fish~~
- ~~Meows~~
- ~~Whiskers~~
- Fur

Relevant Characteristics

- Four Legs
- Tail
- Two Eyes
- Fur

Examples

- Created from the **Specific Characteristics**



- Created from the **Relevant Characteristics**
  Note: Building from the model created from eliminating the **Specific Characteristics**, we can still tell that the model is a **cat**.

Using Abstraction For Tasks

Now that we know about filtering out unneeded information, let's take a look at using this with tasks.

If you had a list of instructions for feeding the cat, it might look something like this:

Feed Cat:

- Make sure cat food bowl is empty
- Open cabinet
- Get out cat food
- Fill cat food bowl
- Put cat food away
- Close cabinet

Using Abstraction For Tasks

- Now, we need to feed the dog as well, so we need a list of steps on how to feed the dog. It might look like this:

Feed Dog:

- Make sure dog food bowl is empty
- Open cabinet
- Get out dog food
- Fill dog food bowl
- Put dog food away
- Close cabinet

Using Abstraction For Tasks

Now let's take a look at our two tasks.

Feed Cat:

- Make sure cat food bowl is empty
- Open cabinet
- Get out cat food
- Fill cat food bowl
- Put cat food away
- Close cabinet

Feed Dog:

- Make sure dog food bowl is empty
- Open cabinet
- Get out dog food
- Fill dog food bowl
- Put dog food away
- Close cabinet

They look very similar, don't they?
We should try to abstract out the **Specific Details** to create a generic model for feeding pets.

Using Abstraction For Tasks

Let's identify all of the things that are different between the tasks, chances are, those are the **specific details** that we want to abstract away. Then, we can replace the **Specific,** with the **Relevant** and generic terms.

Feed ~~Cat~~:

- Make sure ~~cat~~ food bowl is empty
- Open cabinet
- Get out ~~cat~~ food
- Fill ~~cat~~ food bowl
- Put ~~cat~~ food away
- Close cabinet

Feed ~~Dog~~:

- Make sure ~~dog~~ food bowl is empty
- Open cabinet
- Get out ~~dog~~ food
- Fill ~~dog~~ food bowl
- Put ~~dog~~ food away
- Close cabinet

Feed Pet:

- Make sure pet food bowl is empty
- Open cabinet
- Get out pet food
- Fill pet food bowl
- Put pet food away
- Close cabinet

Using Abstraction For Tasks

Now that we got rid of the **Specific Characteristics**, we can use our new task to show how to feed any of the household animals, instead of needing a task for each one!

Recap:

Abstraction is getting rid of **Specific Characteristics** while keeping the **Relevant Characteristics** needed for completing a task. We do this so we can use the same set of instructions to complete many different tasks, instead of having to create a new set of instructions for every different thing.