LN #13
(1 Hr)

Decomposition, Pattern Recognition & Abstraction

CTPS 2018

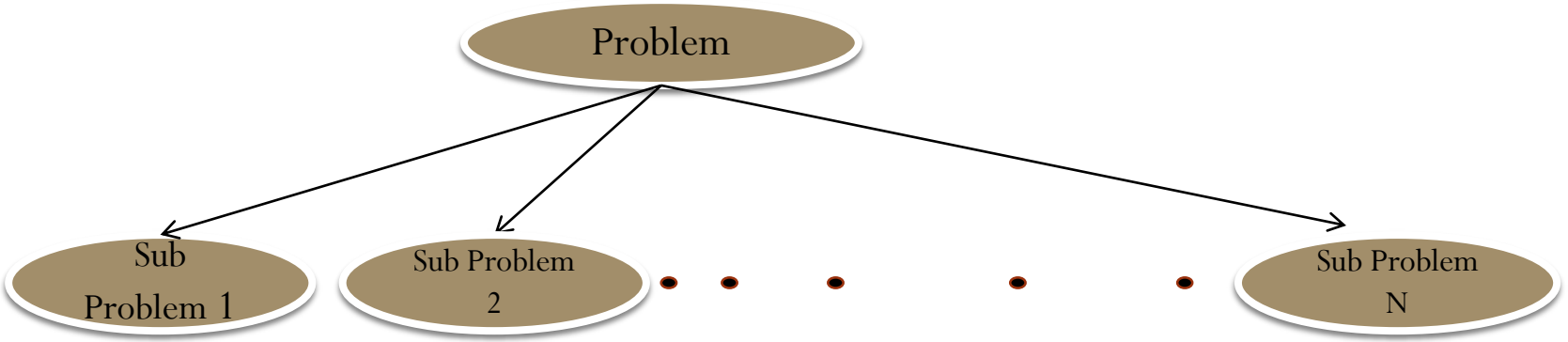
Before computers can solve a problem, the problem and the ways in which it can be resolved must be understood.

Computational Thinking in Practice

- A complex problem is one that, at first glance, we don't know how to solve easily.
- Computational thinking involves taking that complex problem and breaking it down into a series of small, more manageable problems (**decomposition**).
- Each of these smaller problems can then be looked at individually, considering how similar problems have been solved previously (**pattern recognition**) and focusing only on the important details, while ignoring irrelevant information (**abstraction**).
- Next, simple steps or rules to solve each of the smaller problems can be designed (**algorithms**).
- Finally, these simple steps or rules are used to **program** a computer to help solve the complex problem in the best way.

Decomposition

- Decomposition helps by breaking down complex problems into more manageable parts.
- Decomposition is one of the four cornerstones of Computer Science.
- It **involves breaking down a complex problem or system into smaller parts that are more manageable and easier to understand**.
- The smaller parts can then be examined and solved, or designed individually, as they are simpler to work with.



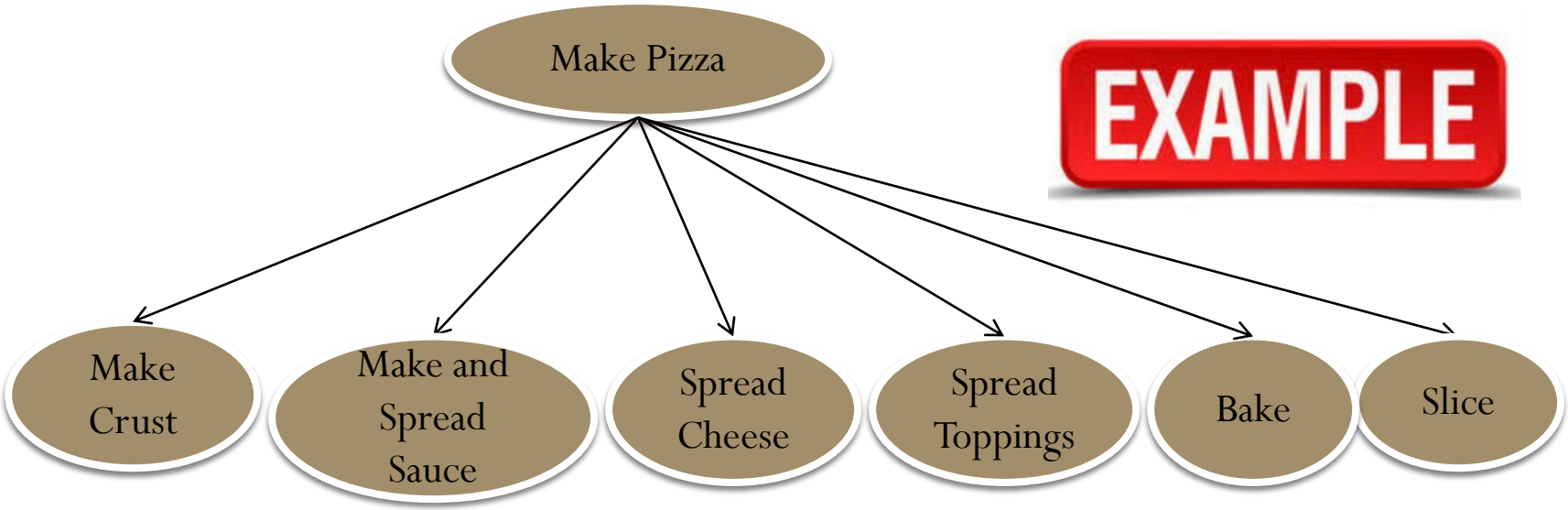
Why is Decomposition important?

- If a problem is not decomposed, it is much harder to solve.
- Dealing with many different stages all at once is much more difficult than breaking a problem down into a number of smaller problems and solving each one, one at a time.
- Breaking the problem down into smaller parts means that each smaller problem can be examined in more detail.
- Similarly, trying to understand how a complex system works is easier using decomposition.

For example:

understanding how a bicycle works is more straightforward if the whole bike is separated into smaller parts and each part is examined to see how it works in more detail.

- The process of making pizza can be split into six sub-problems :



- Each sub problem is at (roughly) the same level of detail.
- Each sub problem can be solved independently.
- The solutions to the sub problems can be combined to solve the original problem.

✓ *Do you see decomposition in your life? ?*

✓ *If yes, what are they??*

Decomposition in real life

General problem ---- sub-problems

- Human body ---------- parts and its role
- Family ------- individual responsibilities
- Organization -------departments
- Chemicals compounds ----- product molecules/elements/simple compounds
- Numbers --- digits and its position
- Velocity --- directions
- Daily dish ------- ingredients
- So on…

It is only normally when we are asked to do a new or more complex task that we start to think about it in detail – to ***decompose*** *the task.*

Imagine that a crime has been committed.
Solving a crime can be a very complex problem as there are many things to consider.

- For example, a police officer would need to know the answer to a series of smaller problems:
  - what crime was committed
  - when the crime was committed
  - where the crime was committed
  - what evidence there is
  - if there were any witnesses
  - if there have recently been any similar crimes
- The complex problem of the committed crime has now been broken down into simpler problems that can be examined individually, in detail.

Imagine that you want to create your first app. This is a complex problem - there are lots of things to consider.

- How would you decompose the task of creating an app?
- To decompose this task, you would need to know the answer to a series of smaller problems:
  - what kind of app you want to create
  - what your app will look like
  - who the target audience for your app is
  - what your graphics will look like
  - what audio you will include
  - what software you will use to build your app
  - how the user will navigate your app
  - how you will test your app
  - where you will sell your app

Imagine you want to organize all your DVDs alphabetically and you have a lot of them! Where would you start?

You might decompose the task into the following steps:

1. Take all the DVDs off your shelf.
2. Sort the DVDs into piles based on the first letter of the title.
3. Start with the 'A' pile. Organize this group into alphabetical order by second and third letters.
4. Place them on the shelf.
5. Repeat for the rest of the alphabet.

Imagine you want to create a Computer Game

You might decompose the task into the following steps:

- Formulate the idea
- Create graphics
- Program each sprite one at a time
- Test your game
- Debug and make improvements

The game problem is decomposed into small task

Steps in decomposing the equation of finding square roots of a quadratic equation

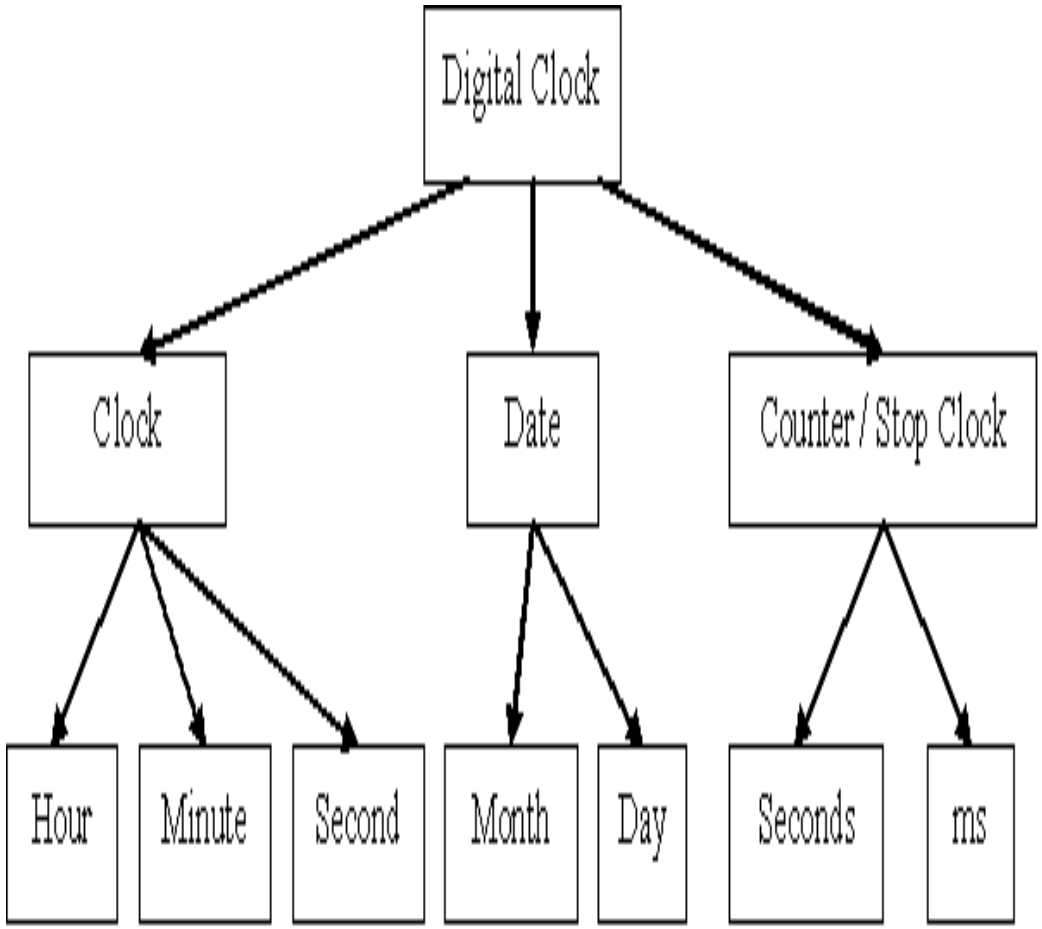
Let's look at an example, the equation to work out the roots of a quadratic equation.

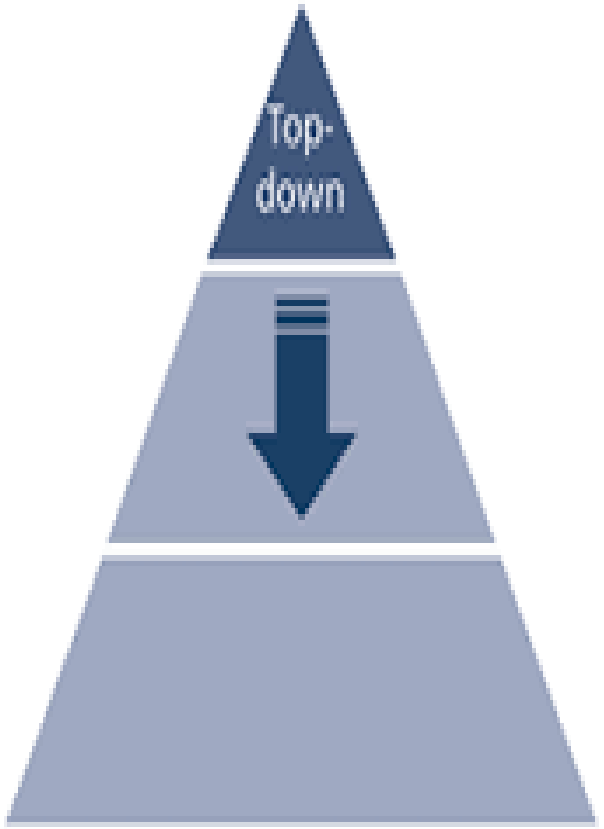
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

On first look it might appear a little scary, but if we decompose it we should stand a better chance of solving it:

1. b^2
2. $4ac$
3. $b^2 - 4ac$
4. $\sqrt{b^2 - 4ac}$
5. $-b + \sqrt{b^2 - 4ac}$
6. $2a$
7. $x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$
8. repeat for $-b - \sqrt{b^2 - 4ac}$

Decomposition – A Top Down approach





Pattern Recognition

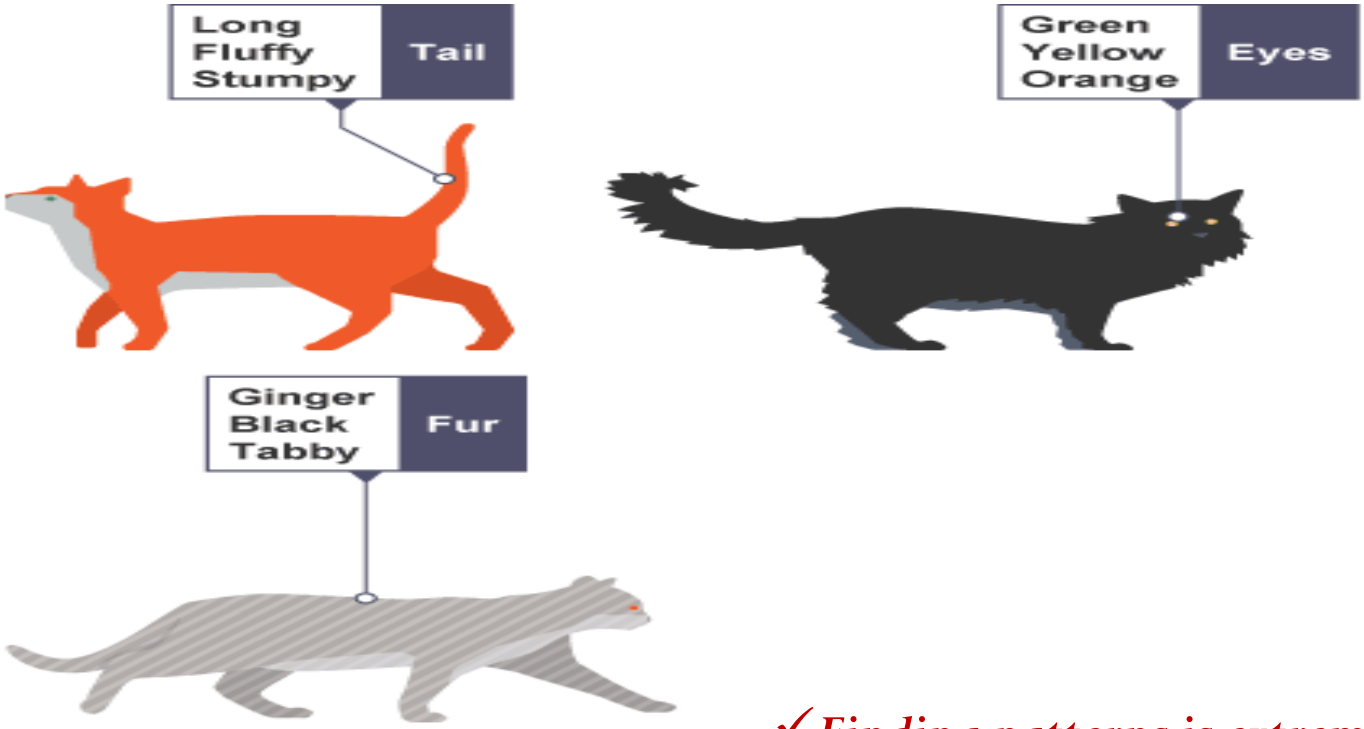
- Once we have decomposed a complex problem, it helps to examine the small problems for similarities or 'patterns'.
- These patterns can help us to solve complex problems more efficiently.
- When we decompose a complex problem we often find patterns among the smaller problems we create.
- The patterns are similarities or characteristics that some of the problems share.
- Pattern recognition is one of the four cornerstones of Computer Science.
- It **involves finding the similarities or patterns among small, decomposed problems that can help us solve more complex problems more efficiently**.

What are patterns?

Imagine that we want to draw a series of cats.

- All cats share common characteristics.
- Among other things **they all have eyes, tails and fur**.
- They also like to eat fish and make meowing sounds.
- Because we know that all cats have eyes, tails and fur, we can make a good attempt at drawing a cat, simply by including these common characteristics.
- In computational thinking, these characteristics are known as patterns.

- **Once we know how to describe one cat we can describe others, simply by following this pattern.**
- The only things that are different are the specifics:
  - one cat may have green **eyes**, a long **tail** and black **fur**
  - another cat may have yellow **eyes**, a short **tail** and striped **fur**



✓ *Finding patterns is extremely important*

Why are finding patterns important?

- Patterns make our task simpler.
- Problems are easier to solve when they share patterns, because we can use the same problem-solving solution wherever the pattern exists.
- The more patterns we can find, the easier and quicker our overall task of problem solving will be.
- If we want to draw a number of cats, finding a pattern to describe cats in general, eg they all have eyes, tails and fur, makes this task quicker and easier.
- We know that all cats follow this pattern, so we don't have to stop each time we start to draw a new cat to work this out.
- From the patterns we know cats follow, we can quickly draw several cats.

What happens when we don't look for patterns?

Suppose we hadn't looked for patterns in cats,

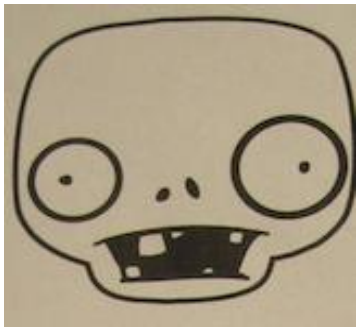
- Each time we wanted to draw a cat, we would have to stop and work out what a cat looked like.
- This would slow us down.
- We could still draw our cats - and they would look like cats - but each cat would take far longer to draw.
- This would be very inefficient, and a poor way to go about solving the cat-drawing task.
- In addition, if we don't look for patterns we might not realise that all cats have eyes, tails and fur.
- When drawn, our cats might not even look like cats.
- In this case, because we didn't recognise the pattern, we would be solving the problem incorrectly.

Recognizing Patterns

- To find patterns in problems we look for things that are the same (or very similar) in each problem.
- It may turn out that no common characteristics exist among problems, but we should still look.
- Patterns exist **among different problems** and **within individual problems**.
- We need to look for both.

Draw a Monster

- Given following monster head(first name) with features(last name). *Can we write a set of instructions for a computer in order to draw a monster's head with creating new monster catalogue?*



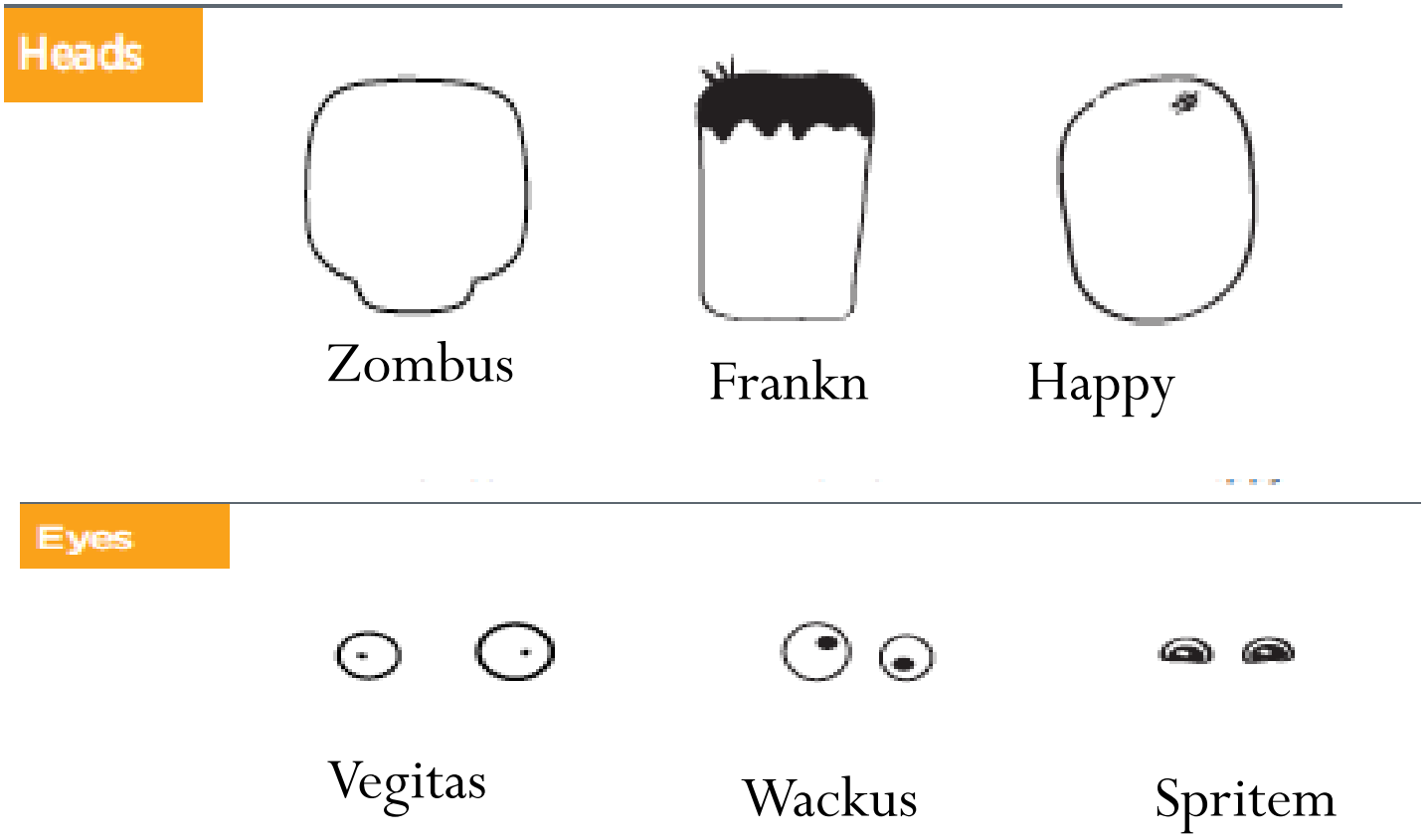
Zombus Vegitas

Frankn Wackus

Happy Spritem

- *Decomposition:*
  - Break drawing process into smaller tasks.
  - Create a list of things needed to be done.
  - Use the list below to start. Can you think of other tasks to add to the list?
    - Example List
      - Create three different monsters.
      - Sort monsters by face shape.
      - Look for similarities in those monsters.
      - Make a list of features to identify.
      - Use identified features to create a new monster.
      - Describe your new monster to your teammates in a step-by-step way and let them try to put it together.

- *Pattern Location*: We can see that all monsters have a head, eyes, ears, nose and mouth.



Ears

Vegitas Wackus Spritem

Nose

Vegitas Wackus Spritem

Mouth

Vegitas Wackus Spritem

E- Mail

- Compose, Inbox, Outbox, Sent mails
- You don't know/need information or details of the internal working to compose mails, send and receive mails, manage Inbox and Outbox.

Tv Remote

- Remote is a interface between user and TV.
- User does not need any information about internal ci

of the remote to use it.

Mobile Phone

- You can dial a number using keypad buttons.
- You don't know/need information or details of the internal working to and receive calls, send and receive messages

Abstraction

- Once we have recognised patterns in our problems, we use abstraction to gather the general characteristics and to filter out of the details we do not need in order to solve our problem.
- Abstraction is one of the four cornerstones of Computer Science.
- It involves filtering out – essentially, ignoring - the characteristics that we don't need in order to concentrate on those that we do.
- It is also the filtering out of specific details.
- From this we create a representation (idea) of what we are trying to solve.
- By *abstraction* we mean anything with multiple "levels", where upper levels hide details of lower levels.

He is my employee

He is my patient

✓ *It involves filtering out…..*

- Doctor views a person by weight, height, age, blood group, symptoms, history of disease, etc.

- Employee views a person name, age, health, degree of study, work experience etc

Model

- Abstraction allows us to create a general idea of what the problem is and how to solve it.
- The process instructs us to remove all specific detail, and any patterns that will not help us solve our problem.
- This helps us form our idea of the problem. This idea is known as a 'model'.
- If we don't abstract we may end up with the wrong solution to the problem we are trying to solve.

- In **pattern recognition** we looked at the problem of having to draw a series of cats.
- We noted that all cats have general characteristics, which are common to all cats, eg eyes, a tail, fur, a liking for fish and the ability to make meowing sounds.
- In addition, each cat has **specific characteristics**, such as **black** fur, a **long** tail, **green** eyes, a love of **salmon**, and a **loud** meow.

These details are known as specifics.

- In order to draw a basic cat, we **do** need to know that it has a tail, fur and eyes.
- These characteristics are relevant.
- We **don't** need to know what sound a cat makes or that it likes fish.
- These characteristics are irrelevant and can be filtered out.
- We **do** need to know that a cat has a tail, fur and eyes, but we **don't** need to know what size and colour these are.
- These specifics can be filtered out.
- From the general characteristics we have (tail, fur, eyes)
- we can build a basic idea of a cat, ie what a cat basically looks like.
- Once we know what a cat looks like we can describe how to draw a basic cat.

- With our cat example, if we didn't abstract we might think that all cats have long tails and short fur.

- Having abstracted, we know that although cats have tails and fur, not all tails are long and not all fur is short.

- In this case, abstraction has helped us to form a clearer model of a cat.

How to perform Abstraction?

- Abstraction is the gathering of the general characteristics we need and the filtering out of the details and characteristics that we do not need.

Example

- When baking a cake, there are some general characteristics between cakes.
  - a cake needs ingredients
  - each ingredient needs a specified quantity
  - a cake needs timings
- When abstracting, we remove specific details and keep the general relevant patterns.

General patterns	Specific details
We need to know that a cake has ingredients	We don't need to know what those ingredients are
We need to know that each ingredient has a specified quantity	We don't need to know what that quantity is
We need to know that each cake needs a specified time to bake	We don't need to know how long the time is

✓ *To find patterns among problems we look for things that are the same (or very similar) for each problem.*

Creating a model

- **A model is a general idea of the problem we are trying to solve.**
- For example, a model cat would be any cat.

  Not a specific cat with a long tail and short fur - **the model represents all cats**.

  From our model of cats, we can learn what any cat looks like, using the patterns all cats share.
- Similarly, when baking a cake, a model cake wouldn't be a specific cake, like a sponge cake or a fruit cake.

  Instead, the model would represent all cakes.

  From this model we can learn how to bake any cake, using the patterns that apply to all cakes.