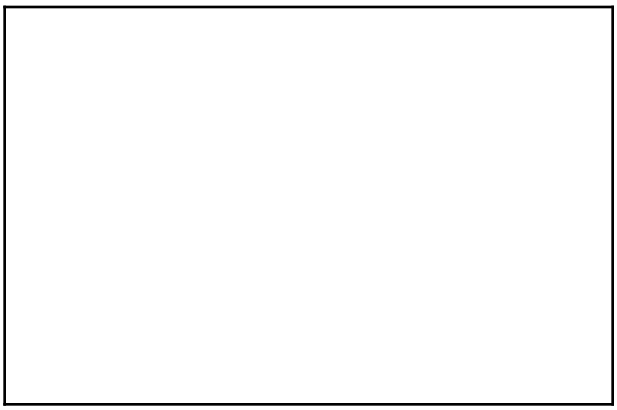
# ABSTRACTIONS

Using this online drawing tool, draw a picture of a chair and paste a screenshot of your result here:



Check off any feature of chairs you ignored when you drew your chair.

| Things I Ignored | |
|---|---|
| Tт Feature | Tт Ignored (Yes/No) |
| Size | |
| Color | |
| Cost | |
| Material | |
| Shape | |
| Brand | |

You created an **ABSTRACTION** of a chair. You did not create a REAL chair! When you designed your abstraction, you included only the features that you considered to be important in solving the problem. You HID or IGNORED any feature that was not important in solving the problem.

### Abstracting in code

We can also create abstractions of real world ideas in code.

In this unit we are creating an abstraction of a **Barmy Farmer**. Ofcourse, farmers have many properties such as

- a name
- hair
- a nose
- a bank account
- a field full of cows

But in the Barmy Farmers game we are going to ignore non-essential details and only focus on essential details!

- a name
- a score
- a bankruptcy status

# Task 1

Watch the video and **follow along** in your IDE.

Answer the following questions:

| | |
|---|---|
| How many attributes/properties does the Farmer class have? | |
| What is the name of the Farmer class's constructor method[1]? | |
| What is the **purpose** of the constructor method? | |
| Write a line of code which will | |

[1] Hint: it's the same as the name of the class!!

| | |
|---|---|
| construct an instance of a Farmer object. His **name** is “Fred”, his **score** is 0 and his **bankrupt status** is false. | |
| What is the identifier of the object that you constructed in the previous line of code? | |
| How does Java solve the problem of **ambiguity** between parameter identifiers and object variable identifiers eg name = name; | |
| Make your Farmer from the previous question speak! | |

Rate your understanding of things you have encountered so far. Cross out/delete as many stars as necessary:



# Task 2

Every time a Farmer speaks, (s)he says "I do not wish to talk to you!" if his/her bankrupt status is true; otherwise (s)he says "My name is ....."

Revisit the Farmer class. How can we code this new Farmer behaviour?

Redesign the Farmer class to create this adaptation.

Find a way to test your design to see that it works as expected.

# Task 3

**Game Prototype**

***Unlucky Sevens*** sees 2 Farmers, **Jack** and **Jill**, competing for points.

Both Farmers start with a score of 20 and a bankrupt status of false.

The game starts and each Farmer's score can go **up** or **down** randomly.

The unlucky number is 7. If a Farmer's score becomes 7, that Farmer's bankrupt status becomes true and the game ends.

Finally, both Farmers speak and the winner's name is displayed!.

Code this prototype. Note: *you shouldn't have to change the Farmer class*.

Hint: think about the game loop control condition:

```java
//game loop
while(famer1.score != 7 && farmer2.score != 7){
     //play game!
}
```

**SUBMIT YOUR .java file and this worksheet as instructed.**