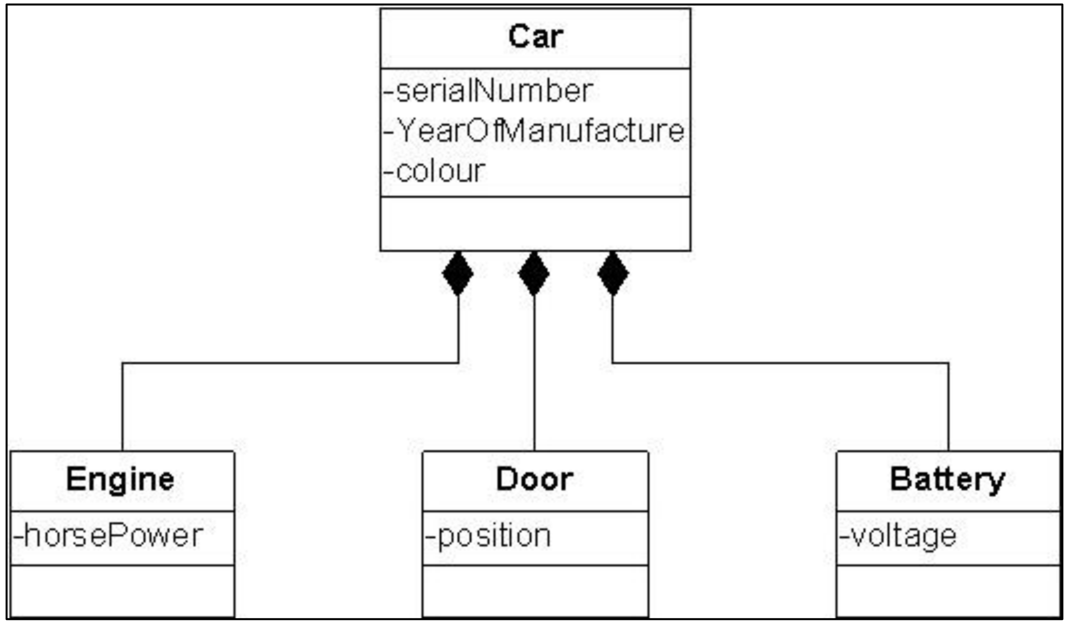
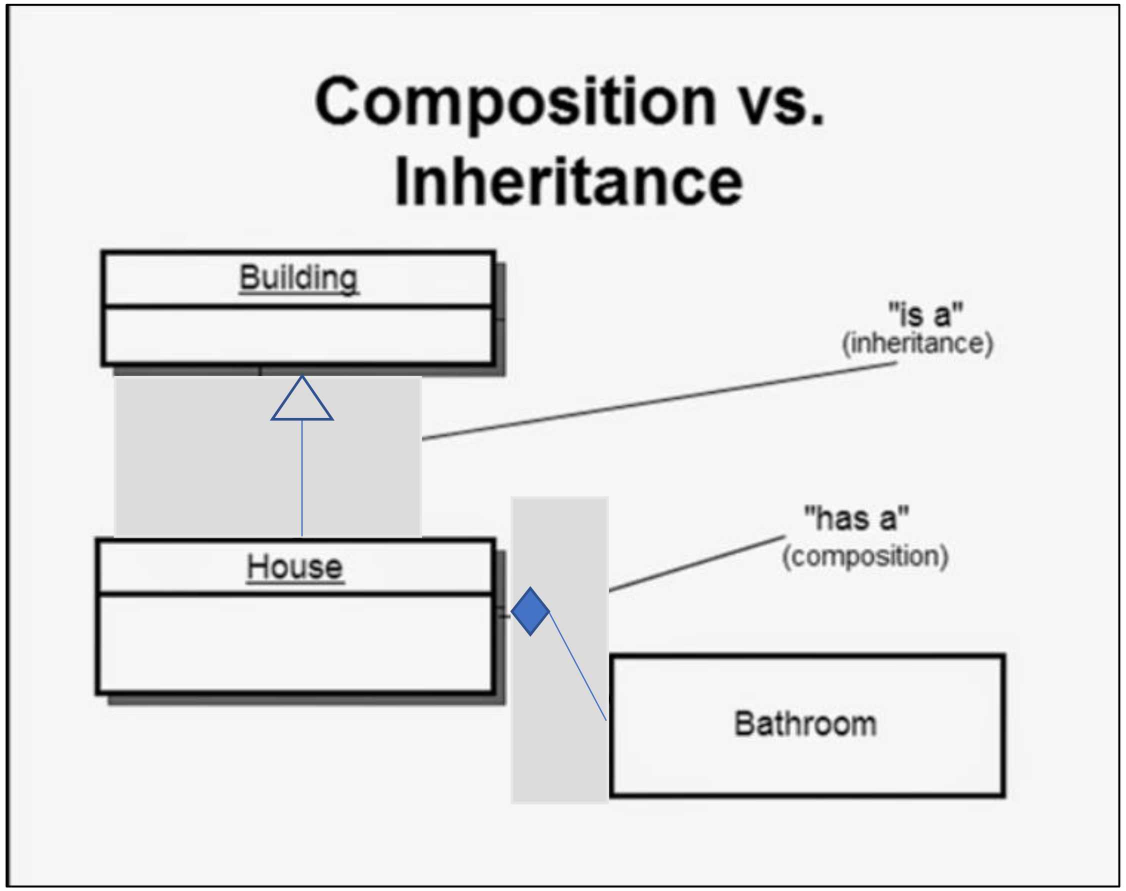
Composition

- **Composition** is the design technique to implement has-a relationship in classes. We can use **java** inheritance or Object **composition** for code reuse. **Java composition** is achieved by using instance variables that refers to other objects.
- For example, a Person has a Job.





Recall exercise1 in Lab 3

- We created a Mammal class, Human class, Doctor class ad Specialization Class.
- Doctor is a Human, Human is a Mammal (These are inheritance)
- However, Specialization is not Doctor.
- Every doctor has a specialization.
- Therefore, we created a composition relationship between Doctor and Specialization.

Example: Every book has an author..

```
public class Author {
// The private instance variables
   private String name;
   private String email;
   private char gender;   // 'm' or 'f'


   // The constructor
   public Author(String name, String email, char
gender) {
      this.name = name;
      this.email = email;
      this.gender = gender;
   }
```

```
 // The public getters and setters for the private instance
variables.
 public String getName() {
      return name;
   }
public char getGender() {
      return gender;
   }
  public String getEmail() {
      return email;
   }
  public void setEmail(String email) {
      this.email = email;
   }

  // The toString() describes itself
  public String toString() {
      return name + " (" + gender + ") at " + email;
   }

}
```

```java
public class Book {
// The private instance variables
  private String name;
  private Author author;
  private double price;
  private int qty;
   // Constructor
  public Book(String name, Author author, double
price, int qty) {
    this.name = name;
    this.author = author;
    this.price = price;
    this.qty = qty;
  }
   // Getters and Setters
  public String getName() {
    return name;
  }
```

```java
public Author getAuthor() {
    return author;  // return member author, which is an
instance of the class Author
  }
  public double getPrice() {
    return price;
  }
  public void setPrice(double price) {
    this.price = price;
  }
  public int getQty() {
    return qty;
  }
  public void setQty(int qty) {
    this.qty = qty;
  }

  // The toString() describes itself
  public String toString() {
    return "'" + name + "' by " + author;  // author.toString()
  }
}
```

```
public class TestBook {
 public static void main(String[] args) {
      // We need an Author instance to 
create a Book instance
      Author myauthor = new Author("J.K. 
Rowling", "rowling@somewhere.com", 'f');
      System.out.println(myauthor);  // 
Author's toString()

      // Test Book's constructor and 
toString()
      Book myBook = new Book("Harry 
Porter", myauthor, 39.99, 99);
      System.out.println(myBook);  // 
Book's toString()

      // Test Setters and Getters
      myBook.setPrice(18.88);
      myBook.setQty(88);
      System.out.println(myBook);  // 
Book's toString()
```

```
    System.out.println("book name is: " + myBook.getName());
    System.out.println("price is: " + myBook.getPrice());
    System.out.println("qty is: " + myBook.getQty());

// invoke Author's toString()
    System.out.println("author is: " + myBook.getAuthor());
    System.out.println("author's name is: " + 
   myBook.getAuthor().getName());
    System.out.println("author's email is: " + 
    myBook.getAuthor().getEmail());
    System.out.println("author's gender is: " + 
    myBook.getAuthor().getGender());

    // Using an anonymous Author instance to create a Book 
instance
    Book moreDummyBook = new Book("Java for more dummies",
        new Author("Peter Lee", "peter@nowhere.com", 'm'),
        19.99, 8);
    System.out.println(moreDummyBook);  // Book's toString()
  }
}
```

Output:

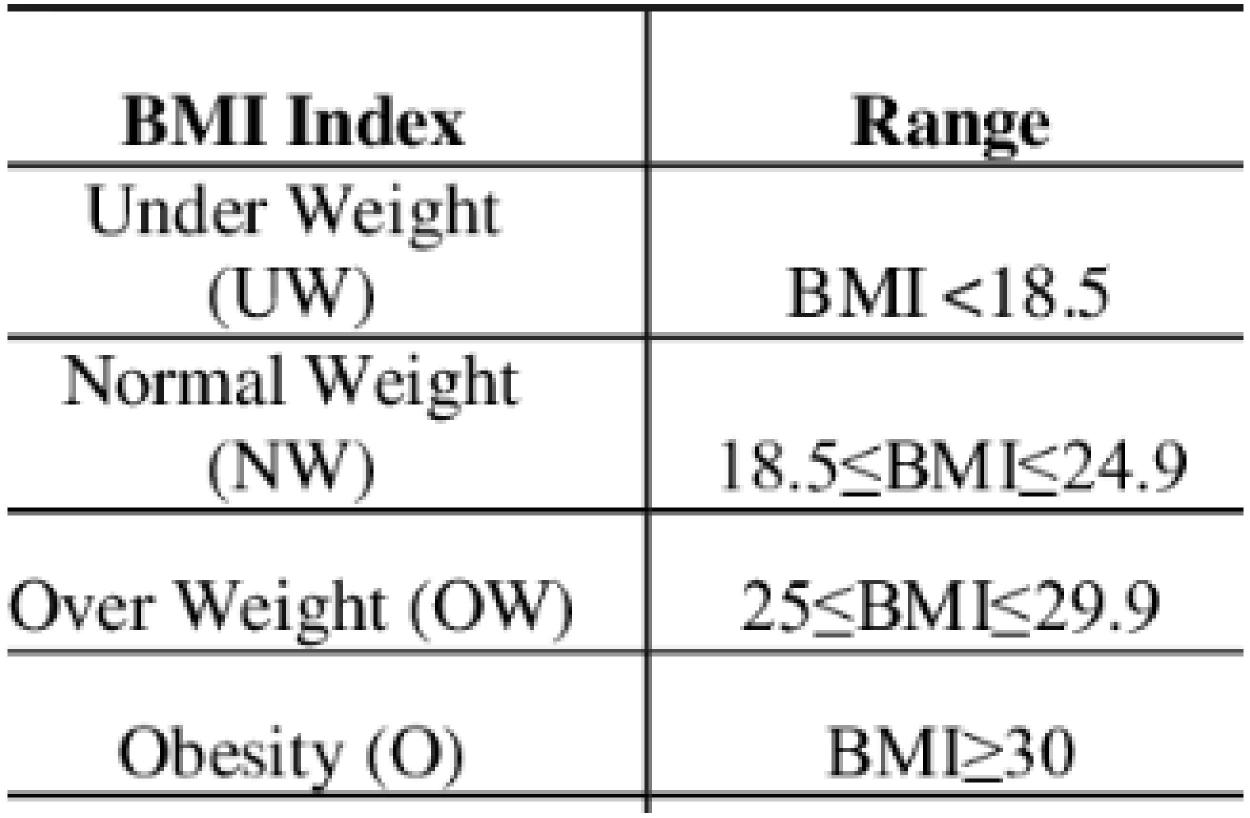
```
J.K. Rowling (f) at rowling@somewhere.com
'Harry Porter' by J.K. Rowling (f) at rowling@somewhere.com
'Harry Porter' by J.K. Rowling (f) at rowling@somewhere.com
book name is: Harry Porter
price is: 18.88
qty is: 88
author is: J.K. Rowling (f) at rowling@somewhere.com
author's name is: J.K. Rowling
author's email is: rowling@somewhere.com
author's gender is: f
'Java for more dummies' by Peter Lee (m) at peter@nowhere.com
```

In-Class Activities:
Every patient has a BMI index

- Create a class called BMI. Create constructor that will take in height and weight and update the BMI index (e.g. BMI = weight/(height*height)).Create getter methods to return height, weight, BMI index and diagnose result such as “Obese”, “underweight or etc. Create a toString method to print all the BMI details.
- Create a Patient class that will take in patient name, patient ID, patient’s BMI, and treatment date. Create the related setter and getter methods and create a toString method that will display the BMI of patient.
- Create a test application to test the composition. You may use the following to invoke constructor:

Patient myPatient = **new Patient("Sponge Bob", "133-577", Bmi, "04/2/2018");**

BMI Reference:



BMI Index	**Range**
Under Weight (UW)	BMI <18.5
Normal Weight (NW)	18.5≤BMI≤24.9
Over Weight (OW)	25≤BMI≤29.9
Obesity (O)	BMI≥30