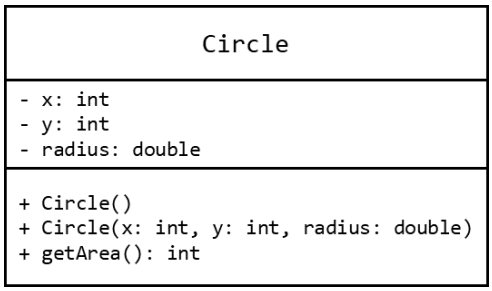
Week 3:

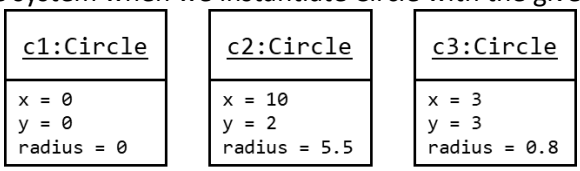
Class diagram for the circle class given the following description

- Attributes(all private):
  - **int x, int y**: represents the location of the circle
  - **double radius**: the radius of the circle
- Constructors:
  - **Circle():** initializes **x, y, radius** to **0**
  - **Circle(int x, int y, double radius):** initializes the attributes to the given values
- Methods:
  - **getArea(): int**



Object diagram for the circle class given

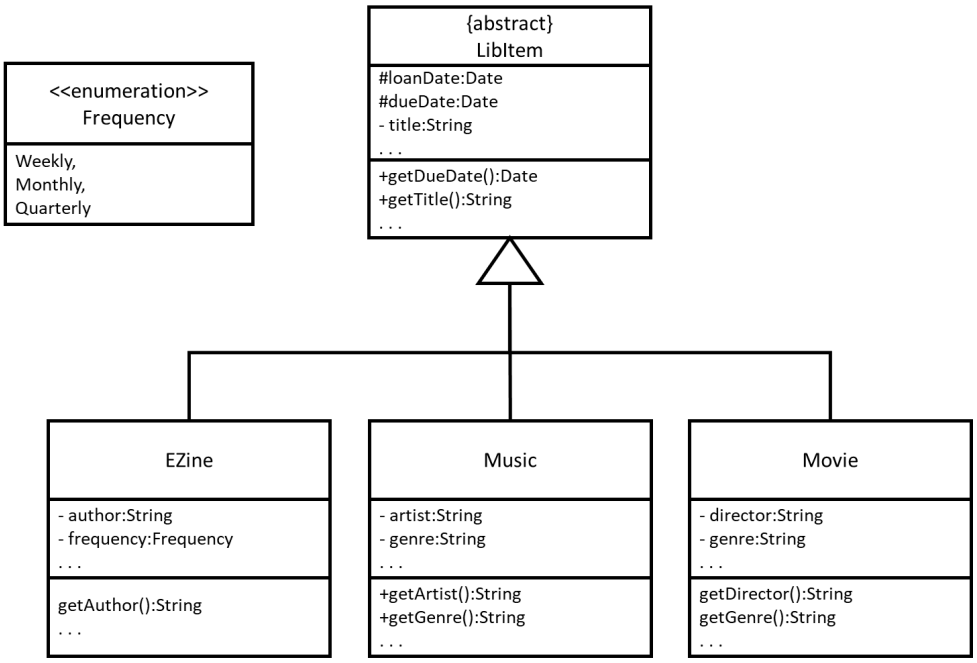
- Runtime state of the system when we instantiate Circle with the given parameters:



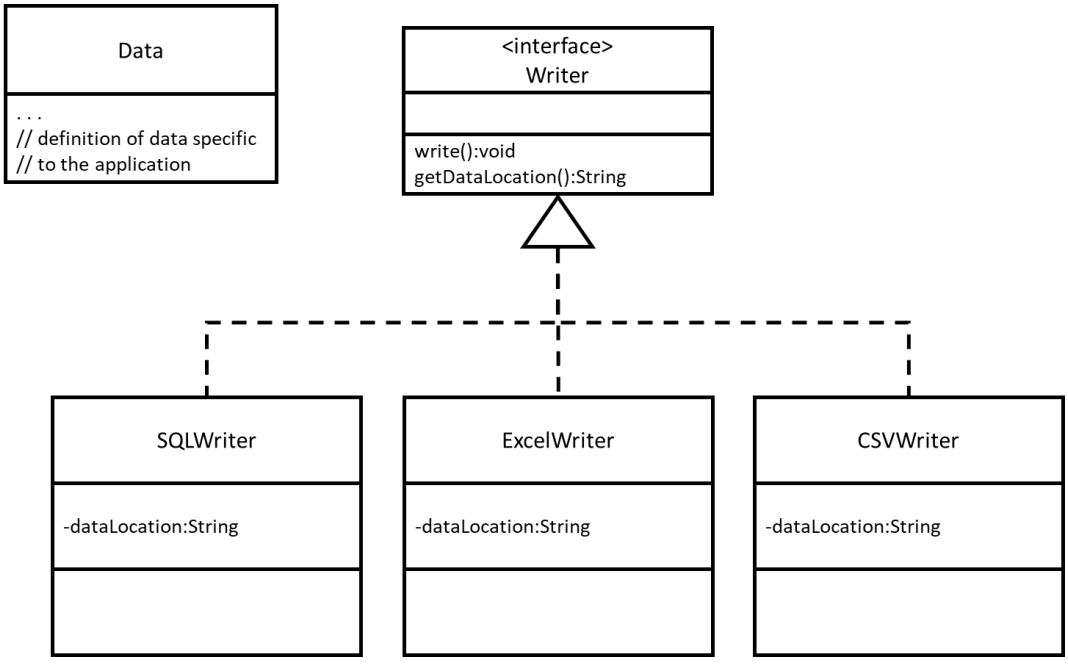
Week 4:

Interfaces and Abstract class

- You are building the back-end software for an online library that loans e-magazines, music and movies. You decide to implement each of these items as classes. As you can expect, there are a few common aspects of loaning e-magazine, music or movies. Identify these common aspects (attributes and methods).



- You are working for a company that specialize in data persistence and storage for software. The current version of your library allows the users to store data in various formats. E.g., into a SQL database for warehousing and analysis, excel sheets for managers and csv files for ease of data transportation. Each part of the client software can use some aspects of your library. Your task is to provide a consistent view of the library features to the users.

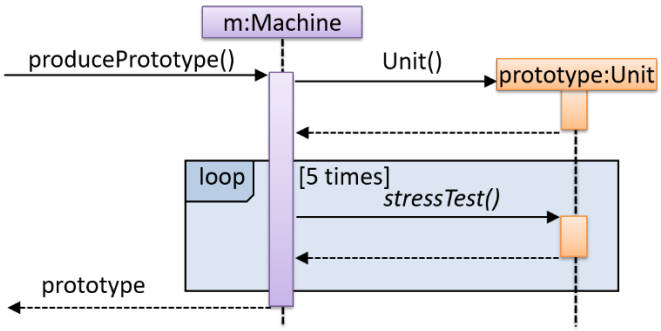


Week 5:

Code review of CLI app

- No header comment for the class
- Import statement – don't import * import specific classes as necessary
- GOODBYE_MESSAGE -> MESSAGE_GOODBYE – To make things consistent
- person -> persons – naming lists/arrays etc.
- Setting userCommand = "nothing" – why this default?
- Boolean exit = false – should be on new line (1)
- Name for exit should be Boolean indicating (2)
- Exit is never set to true in the code.
- loop can be extracted to a method
- Magic strings in the case statement (1)
- Indentation of case statements (2)
- Need a default statement for switch case
- Empty catch block (1)
- Commented out code in the catch block (2)
- Matcher method need not be void (1)
- Naming of matcher (2) – it should be a Boolean, so the name should be Boolean-indicative
- No Javadoc comments for non-trivial methods
- Javadoc comment for showToUser: *Shows*
- Stray comment in showToUser

Explain sequence diagram

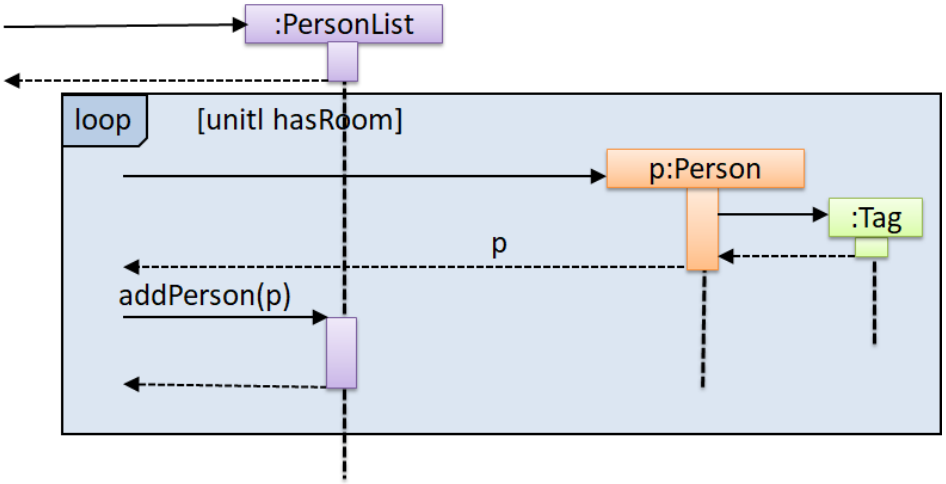


- m is an object of type Machine.
- Instantiate the object of Unit() type -- > by invoking constructor
- m.produceProtype()
- Corner of the box loop important (irrespective of the loop type for/while/do..while, the keyword is loop)
- Arrows need to align with top and bottom of activation bar (which signifies the duration that the object is being called)
- Empty Return Arrow (no return value) -- optional
- Lifeline
- Until 5 times

Draw sequence diagram for the code snippet below.

```
class Person{
    Tag tag;
    String name;

    Person(String personName, String tagName){
        name = personName;
        tag = new Tag(tagName);
    }
}
```

```
class Tag{
    Tag(String value){
        //...
    }
}

class PersonList{
    void addPerson(Person p){
        //...
    }
}
```



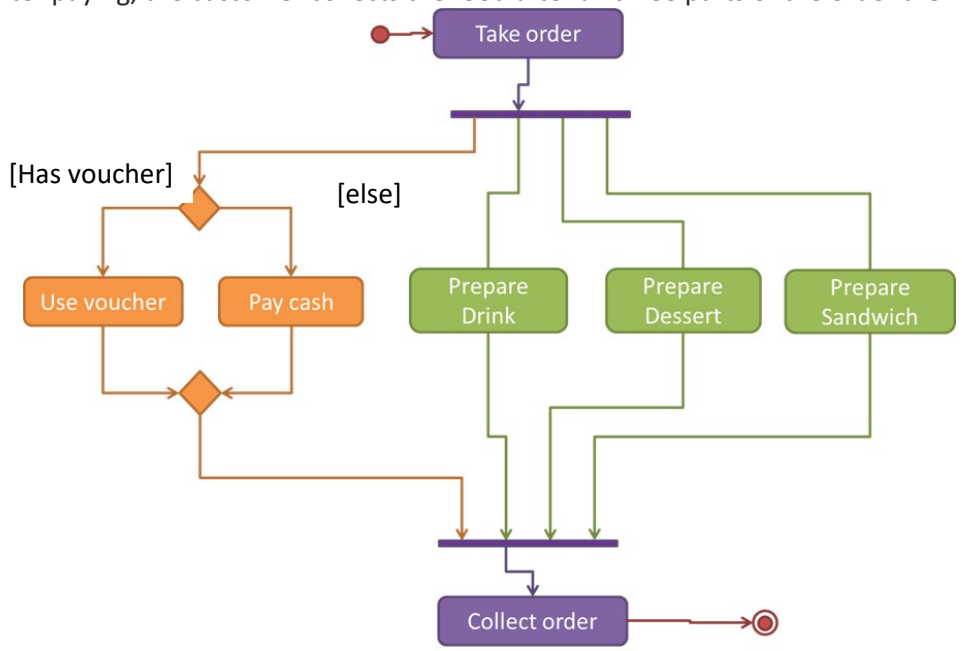
Week 7:

Questions on architecture/api/architectural styles:

- Refer to the text book for solutions.

Model the workflow of the burger shop

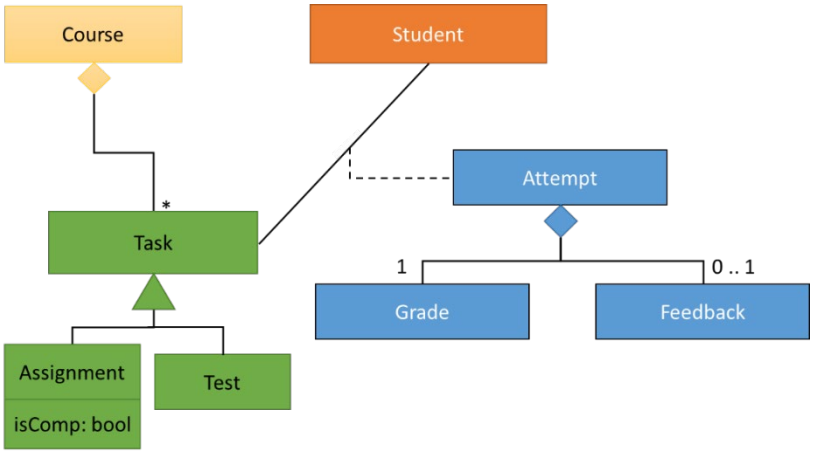
- First, a cashier takes the order.
- Then, three workers start preparing the order at the same time; one prepares the drinks, one prepares the burgers, and one prepares the desserts.
- In the meantime, the customer pays for the order. If the customer has a voucher, she pays using the voucher; otherwise, she pays using cash.
- After paying, the customer collects the food after all three parts of the order are ready.



Week 8

Model the relation between student and tasks:

A course in the university can have a number tasks which can be assignments or tests. Some assignments are compulsory. When a student attempts a task, a grade and an optional feedback is given.



Explain notations of the class diagram

- No solution provided (deemed basic information you should know!)

Draw class diagram for the code:

```java
public interface Billable {
    void bill();
}
```

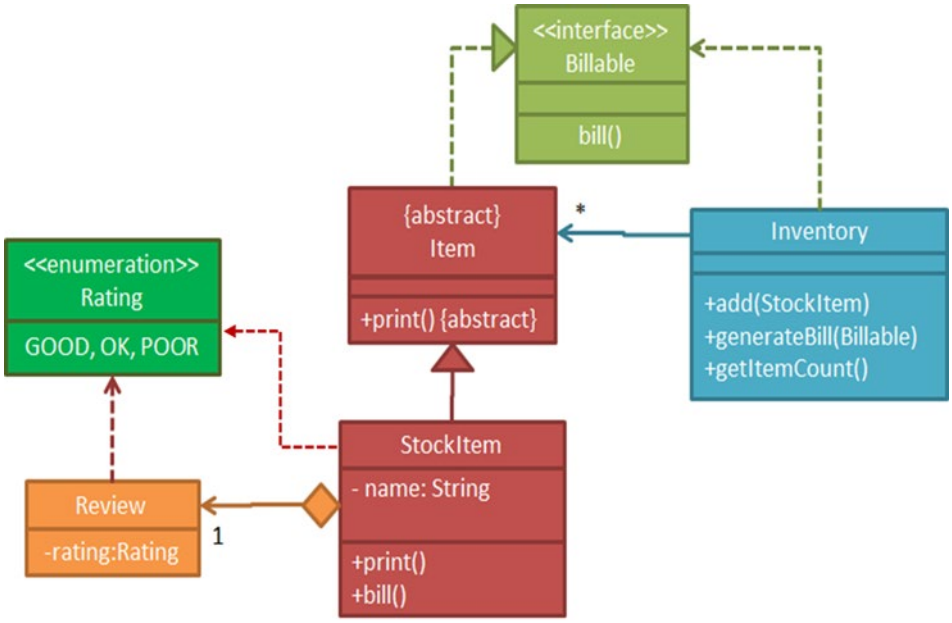
```java
public abstract class Item
         implements Billable {
    public abstract void print();
}
```

```java
import java.util.List;

public class Inventory {
    private List<Item> items;

    public int getItemCount(){
        return items.size();
    }

    public void generateBill(Billable b){
        // ...
    }

    public void add(Item s) {
        items.add(s);
    }
}
```

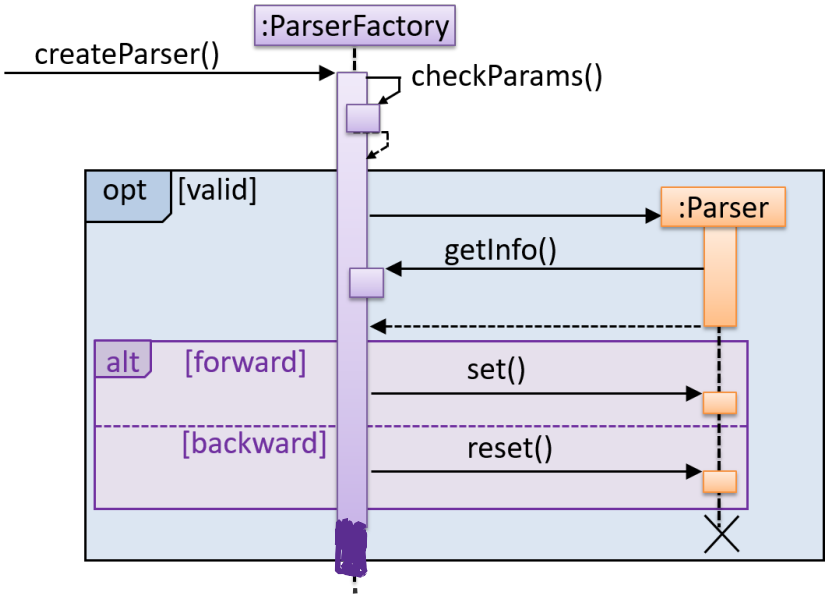
```java
public class Review {
    private final Rating rating;

    public Review(Rating rating) {
        this.rating = rating;
    }
}
```

```java
public enum Rating {
    GOOD, OK, POOR
}
```

```java
public class StockItem extends Item {
    private Review review;
    private String name;

    public StockItem(
        String name, Rating rating){

        this.name = name;
        this.review = new Review(rating);
    }

    @Override
    public void print() {
        //...
    }

    @Override
    public void bill() {
        //...
    }
}
```



Explain sequence diagram – parser factory

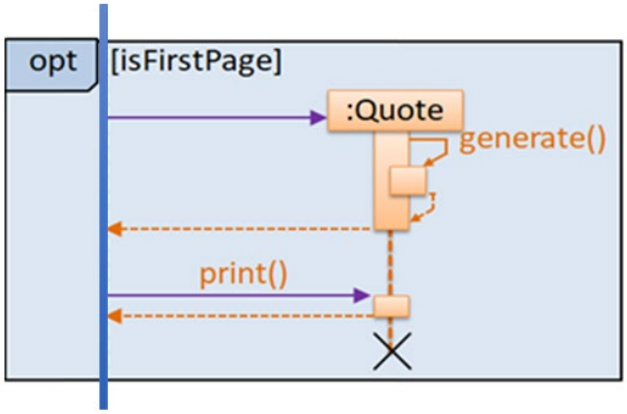


- The activation bar for createParser() should extend beyond the optional box.
- createParser() method of an unnamed ParserFactory object is being called.
- inside the createParser() method,
    - it calls its own checkParams() method.
    - if valid
        - A Parser object is created.
        - The constructor of that Parser objects calls the getInfo() method of the ParserFactory object
        - ParserFactory calls either set() or reset() of the new Parser object based on a condition.
        - Parser object is discarded.
- The entire process is executed only if the condition holds ([params are valid])
- set() or reset() is called depending on the truth value of forward variable

Draw sequence diagram for the snippet:

```
if (isFirstPage) {
    new Quote().print();
}
```

```
class Quote{

    String q;

    Quote(){
        q = generate();
    }

    String generate(){
        // ...
    }

    void print(){
        System.out.println(q);
    }

}
```

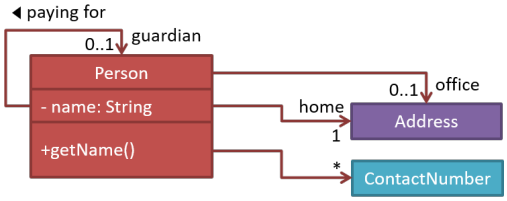


Week 9

Question set 1 – 4

- Basic theory questions that have answers in the textbook. Hence, no answers provided here.

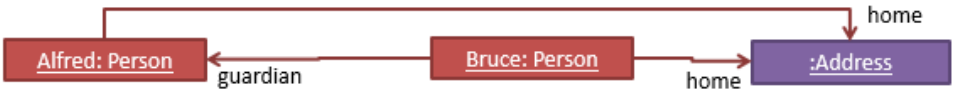
Explain class diagram



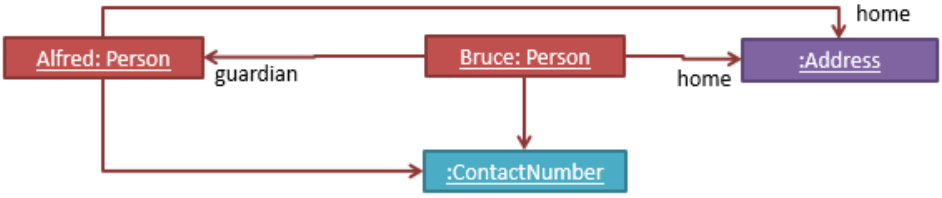
- Class names: Person, Address, ContactNumber
- Private and public members: name(private)
- Associations:
- Navigability: which class knows about which class
- Multiplicity: *, 1, 0..1

Draw object diagram

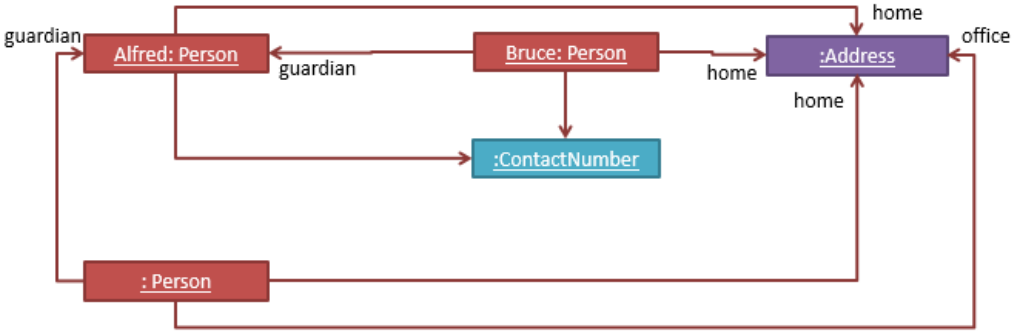
- Step 2: Alfred is the guardian of Bruce.



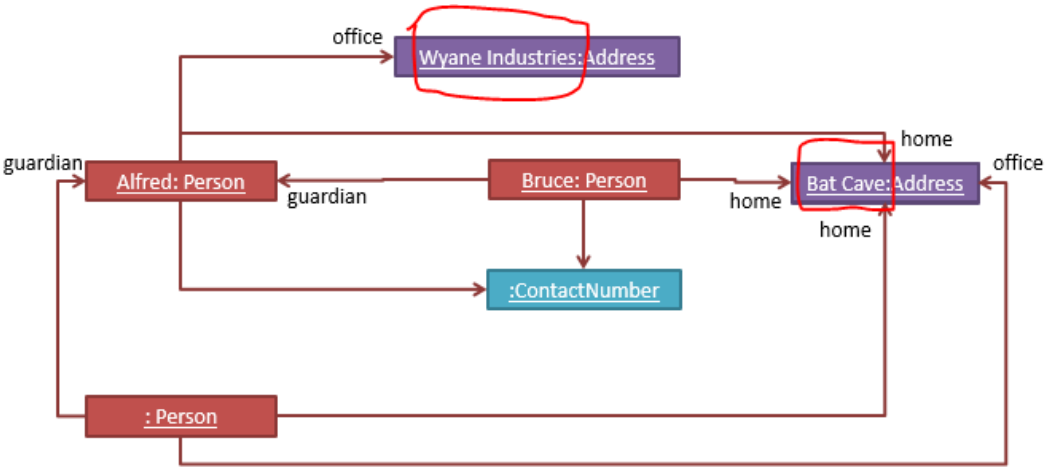
- Step 3: Bruce's contact number is the same as Alfred.



- Step 4: Alfred is also the guardian of another person. That person lists Alfred's home address as his home address as well as office address.



- Step 5: Alfred has a an office address at Wayne Industries building which is different from his home address (i.e. Bat Cave)



Week 10:

All questions are basic theory questions that have answers in the textbook. Hence, no answers provided here.

Week 11:

Test case for `consume` method

SUT: `consume(food, drink)`

Test case	food	drink
TC1	bread	water
TC2	rice	lava
TC3	rock	acid

Test case	food	drink
TC1	bread	water
TC2	rice	water (Any valid value)
TC2.1	rice	lava
TC3.1	rock	water (AVV)
TC3.2	rice (AVV)	acid

Heuristics used: Each valid input in at least one positive test case, No more than one invalid value in one test case

Question set 1 – 4

- All basic theory questions that have answers in the textbook. Hence, no answers provided here