DAY (020-340)
4th Week 20/06/2020

Appointment | Notes | Work to do

Akshatha.Y.E
4AU18EC005

- **Probability:**
  → It is a chance or likelihood of a particular event taking place.
  → An event is an outcome of an experiment.
  → Set of all outcomes of an experiment is called sample space.

- **Types of Probability**
  - A priori Classical Probability.
  - Empirical Probability.
  - Subjective Probability.

$P(A) = m/n$.

where $P(A)$ is always $>= 0$ and always $<= 1$
$P(A)$ is a pure number.

- **Bayes Theorem:**

$$P(B_i|A) = \frac{P(A|B_i)P(B_i)}{P(A|B_1)P(B_1) + P(A|B_2)P(B_2) + \ldots + P(A|B_k)P(B_k)}$$

B_i = i^{th} event of k mutually exclusive and collectively exhaustive events.

A = new events that might impact $P(B_i)$

• Rules for Computing Probability:

1] Addition Rule - Mutually Exclusive Events.

$$P(A \cup B) = P(A) + P(B).$$

2) Events are not Mutually Exclusive.

$$P(A \cup B) = P(A) + P(B) - P(A \cap B).$$

3] Independent Events:

$$P(A \cap B) = P(A) \cdot P(B)$$

4] Events are not independent.

$$P(A \cap B) = P(A) \cdot P(B/A)$$
$$P(A \cap B) = P(B) \cdot P(A/B).$$

• Normal Distribution:

The normal distribution is a continuous distribution looking like a bell. Statisticians use the expression "Bell shaped Distribution".

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left[\frac{(x-\mu)^2}{2\sigma^2}\right]}$$

x is a continuous normal random variable with the property $-\infty < x < \infty$.

• Standard Normal Distribution: $Z = x - \mu / \sigma$

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{[z^2/2]}$$

Appointment | Notes | Work to do

- Natural Ordering:

```
public boolean equals (Object obj) {
    if (this == obj)
      return true;
    if (obj == null)
      return false;
    if (getClass() != obj.getClass())
      return false;
  final Person other = (Person) obj;
    if (nur == null) {
      if (other.name != null)
        return false;
    } else if (!name.equals (other.name))
      return false;
    return true;
  }.
```

- Queues:

```
Queue<Integer> q2 = new ArrayBlockingQueue<Integer>(3);
q2.offer(10);
q2.offer(20);
 if (q2.offer(30) == false){
    System.out.println("offer failed to add third
                        item");
}
for (Integer value : q2) {
    System.out.println("Queue 2 value" + value);
}
```

Appointment | Notes | Work to do

- Using Iterators:

```
Iterator<String> it = animals.iterator();
  while (it.hasNext()) {
    String value = it.next();
    System.out.println(value);

  if (value.equals("cat")) {
    it.remove();
  }
}
  System.out.println();
/// Modern iteration, Java 5 & later.
  for (String animal : animals) {
    System.out.println(animal);
```

- Implementing Iterable:

```
public class App{
  public static void main(String[] args){
    UdLibrary udLibrary = new UdLibrary();

  for (String html: udLibrary){
    System.out.println(html.length());
    System.out.println(html);
  }

  }

}
```

Deciding which collection to Use:

```
Set<String> set1 = new HashSet<String>();
Set<String> set2 = new TreeSet<String>();
Set<String> set3 = new LinkedHashSet<String>();

//////////Maps//////////
Map<String,String> map1 = new HashMap<String, String>();
Map<String,String> map2 = new TreeMap<String, String>();
Map<String,String> map3 = new LinkedHashMap<String, String>();
}
}
```

- Complex Data Structure:

```
public static String[] vertices {
    "ambulance", "helicopter", "Lifeboat"
};
public static String[][] drivers = {
        {"Fred", "Sue", "Pete"},
        {"Sue", "Richard", "Bob", "Fred"},
        {"Pete", "Mary", "Bob"}
}
}
```