```
class Person
  implements
  Comparable<Person> {

 int age;
 string name;
 double sal;

 // constructor

 public int compareTo
            (Person other) {
    if( sal < other.getSal ()){
         return -1;
    }
    if (sal > other.getSal ()){
         return 1;
    }

    return 0;
 }

}
```

```
plist= {
    [31, Dharma, 50000 ],  → P1
    [23, Rohan, 100000 ],  → P2
    [45, John, 76000.25 ], → P3
    [17, Krishna, 10000]   → P4
}

Collections.sort (plist);

    // plist → {P1, P3, P2, P4}
```

```
public interface Comparator<T> {
    public int compare(T t1, T t2);
}
```

```
public AgeComparator implements
            Comparator<Person> {

    public int compare(Person p1, Person p2){
        if(p1.getAge() < p2.getAge()) {
            return -1;
        }
        if (p1.getAge() > p2.getAge()) {
            return +1;
        }
        return 0;
    }
}
```

Collections.sort(plist) $\rightarrow$ p1.compareTo(p2)

Collections.sort(plist, comparatorObj) $\rightarrow$ ComparatorObj.compare(p1, p2)

```
plist = {p1, p2, p3, p4}
AgeComparator ac = new AgeComparator();
Collections.sort(plist, ac);
// plist → {p4, p2, p1, p3}
```

ac.compare(p1, p4) → 1

```
public SalComp imp Comparator<Person>{
    public int compare(Person p1, Person p2) {
        if(p1.getSal() < p2.getSal())
            return -1;
        if(p1.getSal() > p2.getSal())
            return 1;
        return 0;
    }
}
```

```
SalComp sc = new SalComp();
Collections.sort(plist, sc);
// plist → {p1, p3, p2, p4}
```

carsList   class Car implements Comparable{
                    . . .
c1   c2   }

Collections.~~sort~~(carList) $\longrightarrow$ c1.compareTo(c2)

YearComparator yc = new YC()

Collections.sort(carList, yc) $\longrightarrow$ yc.compare(c1, c2)

```
class YearComparator implements
              Comparator<Car> {

    public int compare(Car c1, Car c2) {
          if (c1.getYear() < c2.getYear())
                return -1;
          c1.y > c2.y
                ret +1;
          return 0;
    }
```

```
String a = "Rohan"
String b = "Dharma"

a.compareTo(b)      1
b.compareTo(a)     -1
```

2 4 5 9 8 1

1 | 4 5 9 8 2

1 2 | 5 9 8 4

4 2 5 9 8 1

I

2 4 5 9 8 1

2 4 5 8 9 1

2 4 5 8 1 9

II

2 4 5 1 8 9

I a – 50%

b – 30%

c – 20%

II [a , a , a , a , a , b , b , b , c , c]

```
Random r = new Random(42)
r.nextInt() → 15 , 15 , 15
r nextInt() → 28 , 28 , 28
```

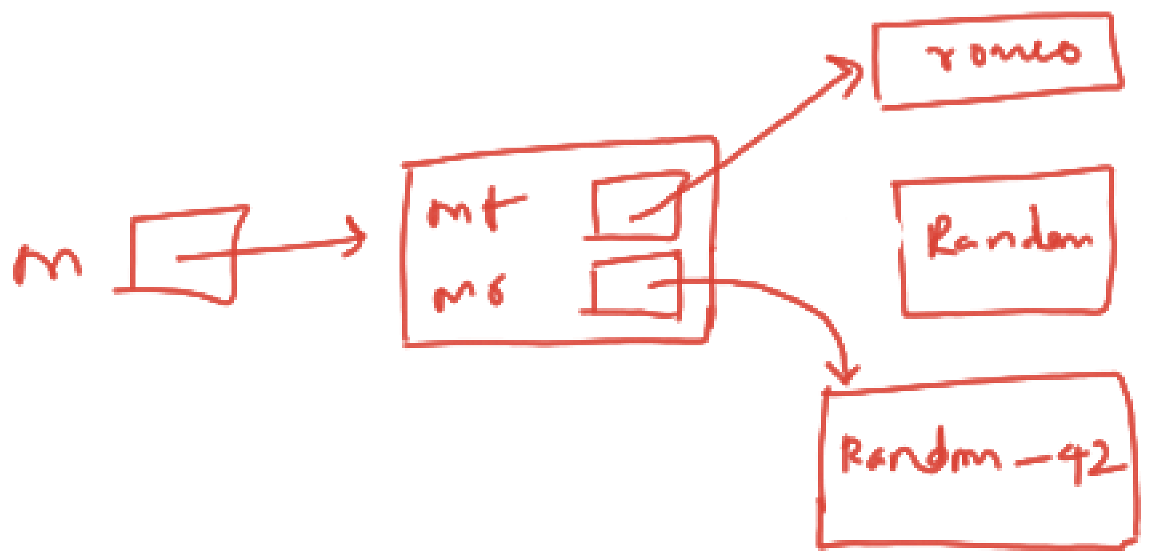
training text= "abcabeadcbea"
0 1 2 3 4 5 6 7 8 9 10 11

myRandom.nextInt(7)

zero-order → 5

b a a c a

MarkovZero m = new MarkovZero();
m.setTraining(r);
m.setRandom(42);



one-order → 7

a b e a d c b

order-two → 10

a b c a b e a d c b

One-order → 5

$\boxed{a}$ b ___ ___ ___

```
Random r1 = new Random();
print(r1.nextInt(10));   // 3
print(r1.nextInt(10));   // 7
print(r1.nextInt(10));   // 2

Random r2 = new Random(101);
print(r2.nextInt(10));   // 2
      "                  // 9
      "                  // 7
```

mt "antre"
0 1 2 3 0

st = 0 key = t

ind = 2 [ ]

mt.ss (0, 3) ant,

0 1 2 3 4
s = "apple"

s.sub(2,4) → "pl"

s.sub(3,3) → ""

s.sub(3,4) → "l"

↓
a t t t r [t,
↑

key = tt

0 1 2 3 4
a b r t t key - tt
↑

5-2

train = "this is a test yes this is a test"
0 1 2 3 4 n-1

markov-zero → 6 chars

h t _ a t s

markov-two → 8 chars

th: [i, i]

h i s _ t h i s

hi: [s, s]

hi: [s, s]

s_: [i, a, t, i]

_t: [e, h, e]

is: [_, _, _, _]

th _ _ _ _ _ _ _ _

rand.nextInt(10) → 0 to 9

sb → hello
0 1 2 3 4

sb. substring (3) → "lo"
sb. substring (4) → "o"

sb → "... there"
$\downarrow$
len "n"

n-2 n-1

n = sb. length( )
sb. ss(n-2) → "re"
sb. ss(n-1) → "e"
⋮
sb. ss(n-k) → last k chars

$$\overset{5\ 6\ 7\ 8\ 9}{\text{"hello}\underbrace{\text{world}}\text{"}}$$

i = r.nextInt(6)

" hello world "   10
  0 1 2 3 4 5 6 7 8 9

$$i = r.next(10-4) \rightarrow \underbrace{0,1,2,3,4,5}$$

key = t.ss(i, i+4);   i = 5 ⇒ word
                      i = 2 ⇒ llow

                      i = 6 ⇒ orld

```
class
MarkovOne {

    public void setTraining( String t){..}

    public String getRanText(int n){
        ...
    }
}
```

```
public class MarkovFour {

    public void setTraining()(
        ..
    }
    public String getRanText(int n){
        ..
    }
}
```

```
runMarkovOne () {
    MO  m = new MO()
    m.setTraining(..);
    m.getRanText(500);
}
```

```
run MarkovFour () {
    MF m = new MF()
    m.setTraining(..);
    m.getRanText(500);
}
```

```
public interface IMarkovModel {
    public void setTraining
                    (String text);
    public String getRanText
                    (int n);
}
```

```
public MT implements
            IMarkovInterface {
    public setTraining(Str t) {
        ...
    }
    public Str getRanText( int n){
        ... // logic to generate
                  order 2 text
    }
    ...
}
```

```
IMarkovModel m0= new MarkovZero();
IMarkovModel m4 = new MarkovFour();
runMarkovModel (m0);
runMarkovModel (m4);
```

```
runMarkovModel(IMarkovModel m)
{
    m.setTraining(..);
    m. genRanText(500);
}
```

```
public abstract class MarkovBaseModel {
      protected String myText;
      protected Random myRandom;

      protected AL<Str> getFollows(Str key) {
              . . .
              . . .
      }

        protected        void setTraining(Str t) {
            myText = t;
        }
        public abstract Str
                getRanText(int numChars);
   }
              Subclass              Base/parent
                 ↑                   ↑   class
public MarkovTwo extends MarkovBaseModel {
    public Str getRanText(int numChars) {
        //logic for order 2 text

        myText
        getFollows(..);

        ...
  }
}
```

text = "this is a text"

this is a text

$\underline{i}\,\underline{s}\ \_\ \underline{i}\,\underline{s}\ \_\ \_$

is $\rightarrow$ [_ , _]

s_ $\rightarrow$ [i, a]

_i $\rightarrow$ [s]

{

th : [i],

hi : [s]

is : [_ , _]

s_ : [i, a]

text = "this is a test" $l = 14$

0 1 2 3 4 ... 12 13

1. $s = 0$, $e = 3$ } $k =$ thi

2. $s = 1$, $e = 4$ } $k -$ his

⋮

n. $s =$, $e = 14$ }

$e <$ length()

text = "this thia"

I. k = thi
   f = s

II k = his
   f = _

:

k = thi
f = a

```
{
  thi : [s, a]
  his : [_]
  is_ : [t]
  s_t : [h]
  _th : [i]
}
```

. . . . . t$\underline{\text{his}}$

```
m =   EfficientModel(3)
m.SetTraining(romeo);

m.generateRandomText(100);


m.setTraining(confucius);
m.generate RandomText(50);
```

"this is a text"

getFollows ("i") → { s, s }

{ "this", "is", "a", "text", "yes", "this", "are", "some" }

getFollows ("this") → { "is", "are" }

$\{ \overset{0}{\text{"this"}}, \overbrace{\text{"this", "this"}}, \text{"a"} \}$

getFollows ("this", "this") → { "this", "a" }

generatedText → ' this is w2
k1 k2

Iit → getF(k1, k2)

[w1, w2, w3]

ind → 1

next → w2

WG wg1 = ["a", "bc", "de"]

WG wg2 = ["a", "bc", "de"]

wg1 == wg2; → false

wg1.equals(wg2);

equals ()

o → wg2

myWords → wg1.myWords

c ["ab", "cde"] "abcde"

o ["abc", "de"] "abcde"

WG wg1 = ["ab", "cd", "de"]

wg1.shiftAdd("hello")

↓

new WG obj

["cd", "de", "hello"]

```
Str[] a = ["a", "b", "c", "d"]
           0    1    2    3

WG w1 = new WG(a, 1, 3); → [b, c, d]
            WG(a, 3, 1); → [d]


          wg = new WG(a, 1, 3);
          ↓
          ["b", "c", "d"]

   -      wg.shiftAdd("hello")
             out → ["c", "d", "hello"]
                    0    1    2
```