⇒ AffineCiphers.

eg:- Keyset - (7, 2)

plaintext - HELLO

ciphertext - ZEBBW

decode text - HELLO

i) explanation :-

Plaintext - (HELLO)

P

abbreviation	
MC →	Multiplicative_Ciphers ⇒ class
AC →	Additive_Ciphers ⇒ class

$T = (P \times K_1) \bmod 26$

```
Mc ac = new Mc();
T = ac.encryption(P, K1)
```

$C = (T + K_2) \bmod 26$

```
Ac mc = new Ac();
C = mc.encryption(T, K2)
```

Cipher text - ZEBBW

→ Additive_cipher - class

•) outline -
default Constructor
→ public Additive cipher ();
1) Initialization of
HashMap map with
of English alphabets
with ASCII values.

eg:-
index i = 97
ch = (char) 97 = 'a'.

•) encryption Method.
public String encryption (String input, int key)

→ converts inPut string to char Array

ch[] = input.toCharArray();

→ check if the character is in Map.
→ validates the character.
→ Gets corresponding key values.
temp = Map.get (ch[i]);

Sum = ((temp + Key) % 26) + 65;

The appends the sum into
out put string

Output + = (char) sum;

Then returns Output String.

2) decryption Method.

public String decryption (String output
, int Key)

→ Convert Ciphertext String to
CharArray.
ch[] = output.toCharArray();

→ Check if the Character is Hash map.
→ validates the Character
→ gets corresponding Key value.
temp = map.get (ch[i]);

There are 2 cases
when temp < Key and temp > Key.

```
sum = ((temp - Key) % 26);
if (sum < 0)
{ sum = 26 + sum; }
```

Why addition of 26?
Ans:- when temp < Key it will give negative values.

eg:- $-15 \bmod 26 = 11$

But java ⇒ $-15 \bmod 26 = -15$

To get 11., you have to add 26

Now - sum = 26 + sum;
Sum = -15 + 26 = 11;
Fix the error.

Then again
sum = sum + 65;
output 2 +=
Now append sum to
output String.

output2 += (char) sum);

Then return output2 String.

Multiplicative Cipher.

→ GetSetElement - class.
get all element in Z26.

→ Multiplicative_Inverse - class
has a HashMap
with Inverse fo of key:

→ Gcd → return gcd (a,b)
and if a,26 gcd is 1
then Multiplicative inverse
exist.

→ Then cal Multiplicative inverse
for all element with gcd equal
to 1.
Then Append into ~~Arra~~ HashMap.
eg: (2,7).

→ Constructor:→
public Multiplication_cipher()
→ Initialization of HashMap.

→ Encryption - Method.
public String encryption (String input, int key)

→ Converts input array to charArray.

→ The validates each char

→ gets corresponding key value
temp = map.get(chC[i]);

→ Cipher = ((temp * key) % 26) + 65;

The appent the char to output string and return.
output = output + ((char) sum);

* Get Key Inverse Method.

→ Gets key inverse using
Inversemap = new MultiplicativeInverse
.inverse(26);

→ Inversekey = Inversmap.get(key);

and calls decryption Method.

decryption (output, inversekey);

→ Decryption Method.

→ Converts input string to Char Array.

→ get corresponding values of from Hashmap

→ rest similar to Additive cipher decryption.

→ return the decoded plain text.