WAP which describes that we can overload static method :

MOLD.java

```
}
* A STATIC METHOD WE CAN DIRECTLY CALL FROM THE NON STATIC METHOD.
* WHILE WORKING WITH METHOD OVERLOADING, TYPE PROMOTION APPLIES AUTOMATICALLY
  a) char -> int -> long -> float -> double
  b) byte -> short -> int -> long -> float -> double

  LPrinting ob1
  void main()
  {
    accept(70, 1.5f);
  }

  public static void accept(float x)
  {
    IO.println("float");
  }

  public static void accept(double x)
  {
    IO.println("double");
  }

  public static void accept(long x)
  {
    IO.println("long");
  }

  MOLD.java
  void main()
  {
    accept();
  }

  public void accept(byte b)
  {
    IO.println("byte");
  }

  public void accept(short s)
  {
    IO.println("short");
  }

  Note : By default It is **int type so we cannot assign to byte OR short (both are below than int)**

  MOLD.java
  void main()
  {
    accept();
    accept(byte(30));
    accept(short(15));
  }

  public void accept(byte b)
  {
    IO.println("byte");
  }

  public void accept(short s)
  {
    IO.println("short");
  }
```

WAP that describes the Type promotion in Overloading :

MOLD.java

```
//char -> int -> long -> float -> double

void main()
{
  accept('A');
}

public void accept(char ch)
{
    IO.println("Char");
}

public void accept(int i5)
{
  IO.println("Int");
}

public void accept(long l5)
{
    IO.println("Long");
}


public void accept(float f5)
{
  IO.println("Float");
}


public void accept(double d5)
{
  IO.println("Double");
}

MOLD.java
void main()
{
  accept('A');
}

public void accept(byte b5)
{
    IO.println("byte");
}

public void accept(short s5)
{
    IO.println("short");
}

Note : char we cannot assign to byte OR short
```

Advantages of Method Overloading :

```
public void accept(char c5)
{
  IO.println("char");
}

private void accept(long l5)
{
  IO.println("long");
}
```

Advantage of Method Overloading :

* The advantage of Method Overloading is, we can provide multiple functionalities by using the same method name. It helps to write the code in a more readable and reusable way.

* Java lang IO class has provided overloaded println() method to provide different functionality.
  public static void println(Object obj) : It is used to print various kind of the data
  public static void println() : It is used to move the cursor in the next line.

Real life examples :

```
Example 1                                   Example 2
public void makePayment(Cash cash)          public void unlockPhone(PIN pin)
{                                           {
}                                           }

public void makePayment(CreditCard card)    public void unlockPhone(Pattern pattern)
{                                           {
}                                           }

public void makePayment(DebitCard card)     public void unlockPhone(Face face)
{                                           {
}                                           }

public void makePayment(UPI upi)
{
}
```

What is a local variable in Java

* If we declare a variable **inside the block OR method OR constructor body** (not as a parameter) then It is called local OR stack OR temporary OR AUTOMATIC variable.
  Example :
  void main()
  {
  {
  int a = 100; //local variable
  IO.println("Block 1 "+a);
  }
  {
  int a = 200; //local variable
  IO.println("Block 2 "+a);
  }
  }

* As far as its accessibility is concerned, A local variable must be used **inside the same method OR block OR constructor body.**
  Example :

```
void main()
{
  {
  int a = 200;
  }
  IO.println("Block "+a); //error cannot find symbol a
}
```

* A local variable must be **initialized by the developer before use because a local variable** doesn't have **any default value.**
  Example :
  void main()
  {
  int a;
  IO.println(a); //error

  int a;
  a = a + 10; //error

  }

* We **cannot** apply any **kind of modifier on local variable except final.**
  Example :
  void main()
  {
  final int a = 100; //only final is applicable
  IO.println(a);
  }

* To Java, all the methods whether static OR non-static will be executed in a **special memory** area called **Stack WHIC which works on LIFO (Last In First Out) basis.**

* In Java, Whenever we call (invoke) any method then internally **ONE STACK FRAME** will be created for each and every method as shown in the program + diagram.

//MethodCallingJava

```
void main()
{
  IO.println("Main method started");//1
  m1();
  IO.println("Main method ended");//5
}

void m1()
{
  IO.println("m1 method started");//2
  m2();
  IO.println("m1 method ended");//4

void m2()
{
  IO.println("Hi I am m2 method");//3
}
```

* All the local & parameter variables are executed from the same method STACK FRAME only so we can access these variable from the same method/construction block body only.

* A local variable must be pre-declared and initialized before use.
  Example
  void main()
  {
  IO.println(a); //error
  int a = 100;
  }

What is Parameter Variable :

* If we declare a variable **as a parameter** in method parameter OR constructor parameter (not inside the body) then It is called Parameter variable.

* The scope of parameter variable is same as local variable that inside the same method OR constructor body.
  Example :
  public void deposit(double amount)
  {
  }

  Parameter Variable

* Parameter variables are also executed in the same **Stack Frame** but It is mainly used to receive the value from the outer block (for 3rd time).