```
interface {

void eat();

walk();
}


class a imple inre{


}


y using interface
1.to achieve abstarction
2.to support functionality of multiple inhereitance
3.to achieve loose coupling




interface games{
void tennis();
}

class competition implements games {
public void tennis(){
sysout(tennis game);
}

psvm {

competition c = new competition();
c.tennis();
}
}


multiple inheritance

interface games{
void tennis();
}

interface visitors{
void watch();
}

class competition implements games,visitors {
public void tennis(){
sysout(tennis game);
}
```

```
public void watch(){
sysout(watching game);
}

psvm {

competition c = new competition();
c.tennis();
c.watch();
}
}
```

interface extends another interface

```
interface games{
void tennis();
}

interface visitors extends games{
void watch();
}

class competition implements visitors {
public void tennis(){
sysout(tennis game);
}

public void watch(){
sysout(watching game);
}

psvm {

competition c = new competition();
c.tennis();
c.watch();
}
}
```

default method in interface

```
interface games{
void tennis();

default void players(){
sysout(tennis players);
}

}
```

```
class competition implements visitors {
public void tennis(){
sysout(tennis game);
}

psvm {
games g = new competition()

c.tennis();
c.players();
}
}

static method
 in interface

interface games{
void tennis();

static void players(){
sysout();
}
}



class competition implements games{
public void tennis(){
sysout(tennis game);
}


class static{
psvm {

games c = new competition();
c.tennis();
sysout(games.players());
}
}
}


nested interface

interface games{
void tennis();

interface compettion {
void play();
}
}
```