What is overloading?

Two methods are said to be overloaded if and only if both having the same name but different argument types

abs(int i)

abs(long l)

abs(float f)

**public void** m1() {}

**public void** m1(**int** i) {}

**public void** m1(**int** i, **int** ii) {}

**public void** m1(**long** l) {}

It is also know are compile time poly morphism or static polymorphism or early binding because method resolution takes care by compiler based on **reference type.**

**What is auto-promotion in Overloading?**

While resolving overloaded methods, if exact matches methods is not available then we won’t get any compile time error immediately. First it will promote argument to the next level and check whether matches is avialable or not. If matched method is available then it will be considered.

If the matched method is not available then compiler promoted argument once again to next level, this process will be continued until all possible promotion. Still if the matched method is not available we will get compile time error. The following are all possible promotions in overloading

**byte--> short -->**

**|**

**-- int --> long --> float --> double**

**|**

char -->

What will happen if the compiler found two matching overloaded methods?

The compiler will give more precedence to for child class argument when compared to parent type argument.

Refer MethodOverloadingTest.java & MethodOverloadingTest1.java

What is ambigious call in overloading?

When two overloaded method can take same type of parameter, and compiler could not able to solve the method resolution, we will get ambigious call issue

For examples refer: MethodOverloadingTest2.java

Method overloadiing w.r.t to var-args

For examples refer: MethodOverloadingTest2.java

Method overloading w.r.t inheritance

In overloading method – resolution will be taken care by referece type. In overloading runtime object wont have any role to play.

For examples refer: MethodOverloadingTest3.java

What is constructor overloading?

Within a class we can declare multiple constructor and all these constructors having same name but different type of argument, hence all these constructor considered as overloaded constructor.

Refer: ConstructorOverloadingTest.java for more details

Does the inheritance and overriding concepts are applicable for constructor?

Nope, only overloading is applicable for constructor.

Does abstract can contion constructor?

Every class in java including abstract class can contain constructor but interface cannot contain contructor.

Is recursive constructor invocation is possible?

Nope, compiler would not allow, but in methods compiler would allow but it normally results in stack overflow error.

Refer: ConstructorOverloadingTest.java for more details

No-args constructor and inheritance

1. If parent class contains any argument constructor then while writing child classes we have to take special care w.r.t constructor

2. Whenever we are writing any argument constructor it is highly recommended to write a no-argument constructor also.

Invalids:

**class** Base1 {

Base1(**int** i) {}

}

**class** Child1 **extends** Base1 {

}

Valids:

1. **class** Base1 {}

**class** Child1 **extends** Base1 {}

2. **class** Base1 { Base() {} }

**class** Child1 **extends** Base1 {}

Exception Handling:

**class** Base1 {

Base1() **throws** IOException{}

}

**class** Child1 **extends** Base1 {

Child1() {**super**();} //Unhandled exception: IOException

}