Multiple Dispatch

It is the language’s mechanism to decide which method to invoke based on the actual runtime type of the passed arguments. Java in contrast links the method at compile time based on the static types.

In statically typed languages, including Java, the biggest difference between dispatch and overloading is that overloading is based on the *static* type of parameters (i.e. the choice of which method is actually called is decided compile-time), while dispatch is based on the *dynamic* types (i.e. the decision is made runtime). (Such languages usually don't support multiple dispatch.)

Let us take an example.

**class** Fruit {

@Override

**public** String toString() {

**return** "Fruit";

}

}

**class** Banana **extends** Fruit {

@Override

**public** String toString() {

**return** "Banana";

}

}

**public** **class** Test1 {

**public** **void** print(Fruit fruit) {

System.***out***.println("Plain fruit ...");

}

**public** **void** print(Banana banana) {

System.***out***.println("This is banana ...");

}

**public** **static** **void** main(String[] args) {

Test1 test1 = **new** Test1();

Fruit fruit = **new** Fruit();

**test1.print(fruit); //Output Plain fruit ...**

Banana banana = **new** Banana();

**test1.print((Fruit) banana); //Output Plain fruit ...**

}

}

Output will be

**Plain fruit ...**

**Plain fruit ...**

To resolve the above the above problem , we rewrite the above program in the following manner.

**package** com.ddlab.rnd.type1;

**class** Vegetable {

@Override

**public** String toString() {

**return** "Vegetable";

}

}

**class** Tomato **extends** Vegetable {

@Override

**public** String toString() {

**return** "Tomato";

}

}

**public** **class** Test2 {

**public** **void** print(Vegetable vegetable ) {

**if**( vegetable **instanceof** Tomato ) {

System.***out***.println("It is Tomato ...");

}

**else** {

System.***out***.println("It is vegetable");

}

}

**public** **static** **void** main(String[] args) {

Test2 test2 = **new** Test2();

Vegetable vegetable = **new** Vegetable();

**test2.print(vegetable); // Output It is vegetable**

Tomato tomato = **new** Tomato();

**test2.print((Vegetable) tomato); // Output It is Tomato ...**

}

}

**Output will be**

**It is vegetable**

**It is Tomato ...**