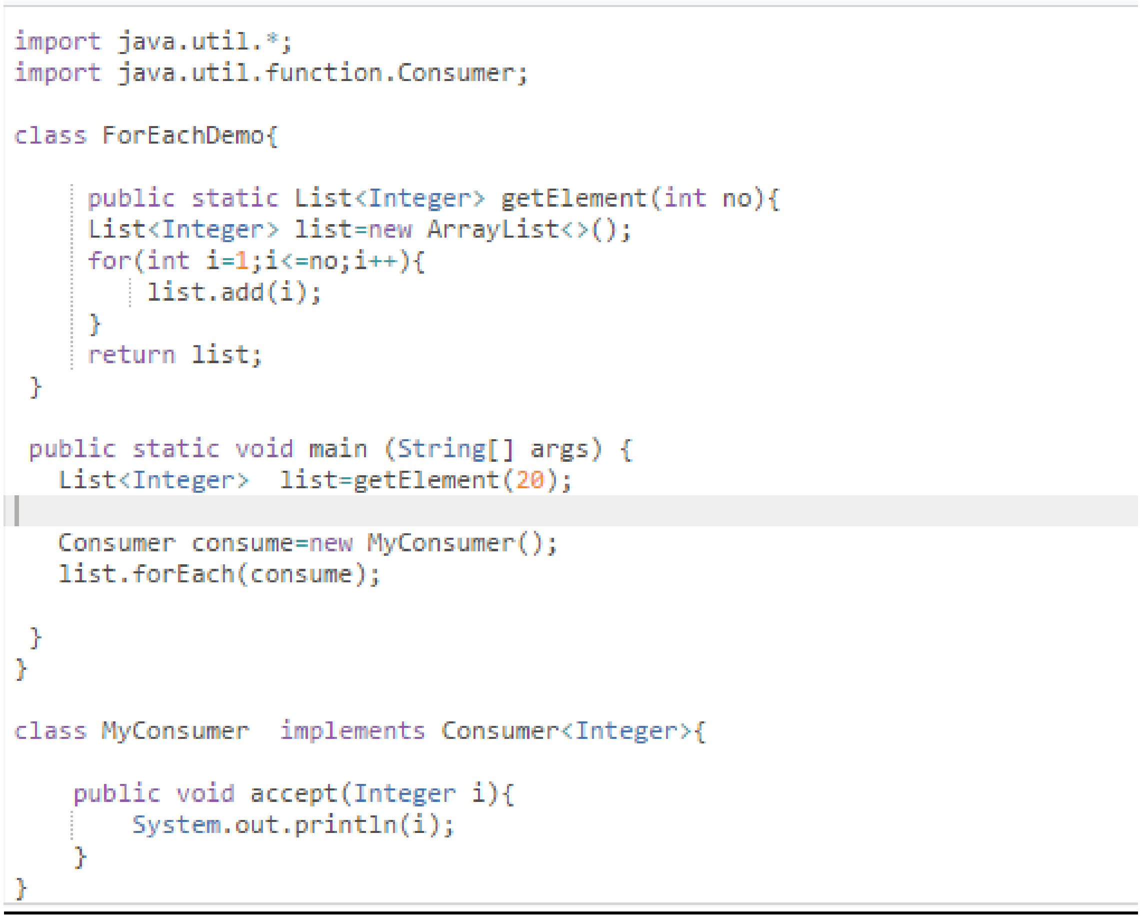
Java 8 Features with Examples

1.forEach () method in Iterable interface:

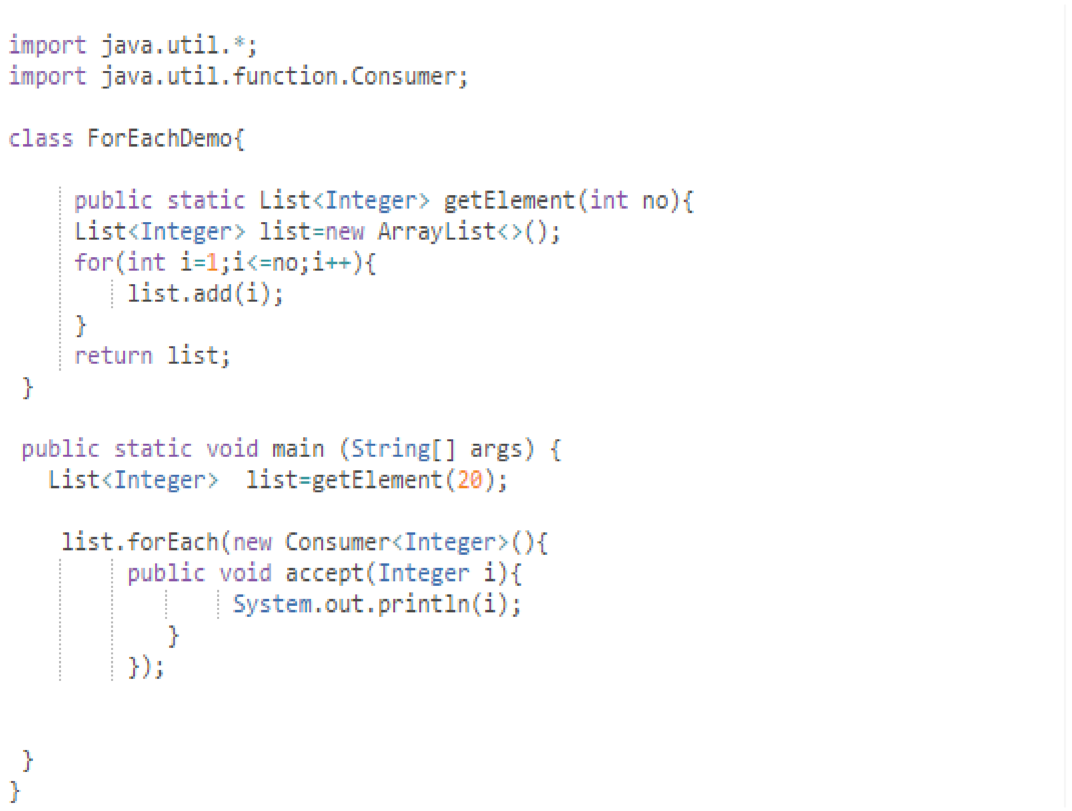
Normally we are using Iterator interface for iterating collection so we might get ConcurrentModificationException if Iterator is not used properly.
Java 8 has introduced forEach method in java.lang.Iterable interface so that while writing code we focus on business logic only. forEach method takes java.util.function.Consumer object as argument, so it helps in having our business logic at a separate location that we can reuse. Let's see forEach usage with simple example.

Example with separate class approach:1



```
import java.util.*;
import java.util.function.Consumer;

class ForEachDemo{

     public static List<Integer> getElement(int no){
     List<Integer> list=new ArrayList<>();
     for(int i=1;i<=no;i++){
         list.add(i);
     }
     return list;
 }

 public static void main (String[] args) {
   List<Integer>  list=getElement(20);

   Consumer consume=new MyConsumer();
   list.forEach(consume);

 }
}

class MyConsumer  implements Consumer<Integer>{

    public void accept(Integer i){
        System.out.println(i);
    }
}
```

Example with anonymous inner class approach:2

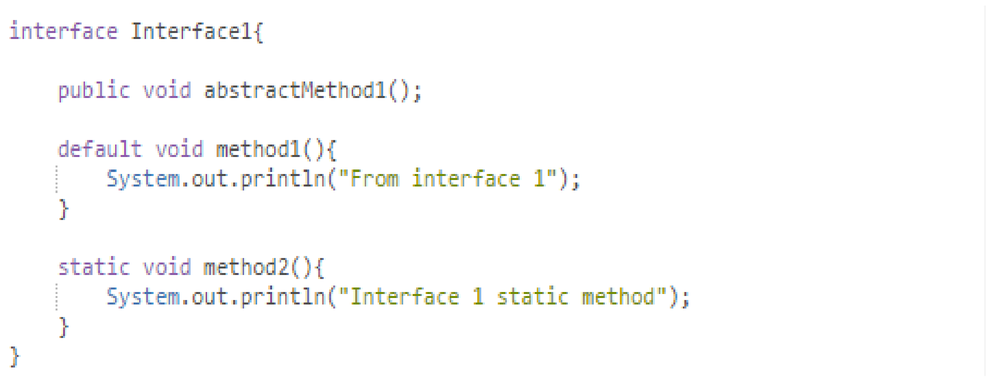

```
import java.util.*;
import java.util.function.Consumer;

class ForEachDemo{

    public static List<Integer> getElement(int no){
    List<Integer> list=new ArrayList<>();
    for(int i=1;i<=no;i++){
        list.add(i);
    }
    return list;
 }

 public static void main (String[] args) {
   List<Integer>  list=getElement(20);

    list.forEach(new Consumer<Integer>(){
        public void accept(Integer i){
            System.out.println(i);
            }
        });


 }
}
```

2. default and static methods in Interfaces:

As we know Java doesn't support multiple inheritance for class level but which can be resolve by interface. If 2 interface contain same method signature also there may be chance for ambiguity to identify by compiler which can be resolved
By default, method in interface.
Let's see one example.
We know that we can't take method body inside interface up to java7 so forEach method is present in Iterable interface is default method with default logic as below.

```
        default void forEach(Consumer<? super T> action) {
        Objects.requireNonNull(action);
        for (T t : this) {
            action.accept(t);
        }
    }
```

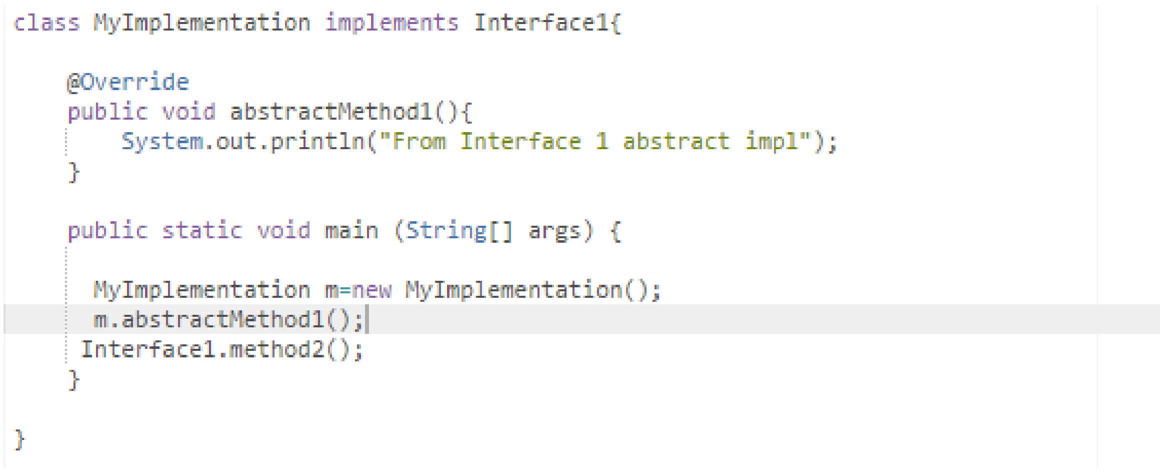
Example to Access Default and static methods.



```
interface Interface1{

    public void abstractMethod1();

    default void method1(){
        System.out.println("From interface 1");
    }

    static void method2(){
        System.out.println("Interface 1 static method");
    }
}
```

In above code interface contains all type of methods like abstract, default and static
NOTE: We can't take object class method as default method in interface like toString(), or hashCode() or equals() it will leads CE.

Let's write implementation class for interface 1.



```
class MyImplementation implements Interface1{

    @Override
    public void abstractMethod1(){
        System.out.println("From Interface 1 abstract impl");
    }

    public static void main (String[] args) {

      MyImplementation m=new MyImplementation();
      m.abstractMethod1();
     Interface1.method2();
    }

}
```

Here no mandatory to override default method and static method.
Normally static method is used in interface to avoid utility class and some common logic like collection sorting etc.…
Default method are used to write your custom implementation logic for specific class.

Note: -
But there is some situation where we have to mandatory to override in Implementation class to avoid diamond problem see below example.

```java
interface Interface1{

    public void abstractMethod1();

    default void method1(){
        System.out.println("From interface 1");
    }

    static void method2(){
        System.out.println("Interface 1 static method");
    }
}


interface Interface1{

    public void abstractMethod1();

    default void method1(){
        System.out.println("From interface 1");
    }

    static void method2(){
        System.out.println("Interface 1 static method");
    }
}
```

```java
class MyImplementation implements Interface1,Interface2{

    @Override
    public void abstractMethod1(){
        System.out.println("From Interface 1 abstract impl");
    }
    @Override
    public void  abstractMethod2(){
        System.out.println("From Interface 2 abstract impl");
    }
    @Override
    public void method1(){
    //U can write any logic for util
        Interface1.method2();
        Interface2.method2();
    }

    public static void main (String[] args) {
      MyImplementation m=new MyImplementation();
      m.abstractMethod1();
      m. abstractMethod2();
      m.method1();
    }

}
```

Here both interface1 and interface2 both contain same default method signature which can be cause ambiguity so in this case we have to override default method in implementation class else it will give CE.

Important points about java interface default methods:

1. Java interface default methods will help us in extending interfaces without having the fear of breaking implementation classes.
2. Java interface default methods has bridge down the differences between interfaces and abstract classes.
3. Java 8 interface default methods will help us in avoiding utility classes, such as all the Collections class method can be provided in the interfaces itself.
4. Java interface default methods will help us in removing base implementation classes, we can provide default implementation and the implementation classes can choose which one to override.
5. One of the major reason for introducing default methods in interfaces is to enhance the Collections API in Java 8 to support lambda expressions.
6. If any class in the hierarchy has a method with same signature, then default methods become irrelevant. A default method cannot override a method from java.lang.Object. The reasoning is very simple, it's because Object is the base class for all the java classes. So even if we have Object class methods defined as default methods in interfaces, it will be useless because Object class method will always be used. That's why to avoid confusion, we can't have default methods that are overriding Object class methods.
7. Java interface default methods are also referred to as Defender Methods or Virtual extension methods.

Important points about java interface static method:

1. Java interface static method is part of interface; we can't use it for implementation class objects.
2. Java interface static methods are good for providing utility methods, for example null check, collection sorting etc.
3. Java interface static method helps us in providing security by not allowing implementation classes to override them.
4. We can't define interface static method for Object class methods, we will get compiler error as "This static method cannot hide the instance method from Object". This is because it's not allowed in java, since Object is the base class for all the classes and we can't have one class level static method and another instance method with same signature.

5. We can use java interface static methods to remove utility classes such as Collections and move all of its static methods to the corresponding interface, that would be easy to find and use.