* **default methods trong interface**
  + Giúp chúng ta dễ dàng mở rộng interface mà không phá vỡ các class được implements từ nó
  + Giúp chúng ta tránh dùng các class tiện ích, ví dụ như tất cả phương thức của class Collections có thể được cung cấp ngay bên trong interface của nó.
  + Giúp chúng ta tháo gỡ các class cơ sở (base class), chúng ta có thể tạo phương thức default và trong class được implement có thể chọn phương thức để override.
  + Một trong những lý do xuất hiện của phương thức default là để nâng cấp Collection API trong Java 8 hỗ trợ cho Lambda Expression.
  + Nếu bất kỳ class nào kế thừa những phương thức default giống nhau, thì nó sẽ không còn hiệu lực. Một điều tương tự, một phương thức default sẽ không thể override một phương thức từ java.lang.Object. Lý do rất đơn giản là bởi vì Object là base class của tất cả các class trong Java. Vì vậy nếu chúng ta có các phương thức của class Object được định nghĩa là phương thức default trong interface, nó sẽ không dùng được bởi vì các phương thức của Object luôn luôn được sử dụng. Đây lý do tại sao chúng ta sẽ không có bất cứ phương thức default nào override các phương thức của class Object.
  + Phương thức default cho phép các Java interface đã tồn tại phát triển thêm mà không gây lỗi trong quá trình biên dịch. Ví dụ như bổ sung các phương thức vào interface java.util.Collection: stream(), parallelStream(), forEach(), …
  + **Static method**
  + Phương thức static là một thành phần của interface, chúng ta có thể sử dụng nó trong class được implements từ nó.
  + Phương thức static rất hữu ích trong việc cung cấp các phương thức tiện ích.
  + Lớp con có thể định nghĩa một phương thức cùng tên với phương thức static của interface cha. Tuy nhiên, chúng ta không thể thêm annotation @Override cho phương thức này.
  + Chúng ta không thể định nghĩa phương thức static của các phương thức thuộc class Object, chúng ta sẽ gặp lỗi “This static method cannot hide the instance method from Object”. Điều này không cho phép trong Java, khi Object là base class cho tất cả các class và chúng ta không thể có một phương thức static và một phương thức khác cùng định dạng.
  + Abstract classes vs interfaces trong Java 8
* Với **default method** trong Java 8, chúng ta thấy rằng chức năng của abstract class và interface giống như nhau. Tuy nhiên điều này không hoàn toàn đúng, mặc dù bây giờ chúng ta có các phương thức cụ thể (các phương thức có thân) trong các interface giống như abstract class, điều này không có nghĩa là chúng giống nhau. Vẫn còn ít sự khác biệt giữa chúng, một trong số đó là abstract class có thể có hàm tạo (constructor) trong khi trong các interface chúng ta không thể có các hàm tạo.
* Mục đích của **interface** là cung cấp sự trừu tượng đầy đủ (100% abstract), trong khi mục đích của **abstract class** là cung cấp một phần trừu tượng hóa (<100% abstract). Điều này vẫn đúng. **Interface** giống như một kế hoạch chi tiết cho lớp của bạn, với việc giới thiệu các phương thức mặc định, chúng ta có thể thêm các tính năng bổ sung trong các interface mà không ảnh hưởng đến các lớp implement hay end user.