پاسخ سوال 4:

متد synchronizedCollection() از کلاس java.util.Collections برای برگرداندن یک Synchronized Collection که توسط collection مشخص شده پشتیبانی میشود، مورد استفاده قرار میگیرد. به منظور تضمین دسترسی سریال ، بسیار مهم است که همه دسترسی ها به Collection پشتیبان از طریق Collection بازگشتی انجام شود. در واقع با استفاده از این متد ما می‌توانیم به Collectionهایی دسترسی داشته باشیم که استفاده از آنها در چند Thread همزمان امن است(به اصطلاح Thread-Safe هستند)

روش کاربرد:

```
ArrayList<String> stringList = new ArrayList<String>();
stringList.add("python");
stringList.add("java");
stringList.add("C++");
stringList.add("C#");
List<String> synchronizedList =
Collections.synchronizedList(stringList);
```

برای دیگر collection ها مانند map نیز می‌توان متد مشابهی را استفاده کرد:

(Collections.synchronizedMap)

به عنوان مثال اگر ما بخواهیم روی یک لیست iterate کنیم باید ابتدا آن را به صورت thread-safe تبدیل کنیم و بعد از آن استفاده کنیم:

```
List<String> uppercasedStringList = new ArrayList<>();

Runnable listOperations = () -> {
    synchronized (synchronizedStringList) {
        synchronizedStringList.forEach((e) -> {
            uppercasedStringList.add(e.toUpperCase());
        });
    }
};
```

پاسخ سوال 5:

در حالتی که چندین interface متدهایی با امضای مشابه داشته باشند. وکلاسی این واسط ها را پیاده سازی کند، یا باید به صورت صریح مشخص کنند که از متد کدام واسط می‌خواهد استفاده کند(با استفاده از super) یا اینکه باید پیاده سازی مخصوص به خودش را داشته باشد.

مثال:

```
interface Clickable{
	default void click(){
		System.out.println("click");
	}

	default void print(){
		System.out.println("Clickable");
	}
}

interface Accessible{
	default void access(){
		System.out.println("access");
	}

	default void print(){
		System.out.println("Accessible");
	}
}

public class Button implements Clickable, Accessible {

	public void print(){
		Clickable.super.print();
		Accessible.super.print();
	}
}
```