Assignment - Beyond Java 8 Features 2

Q1)Create a Record for the Student with the following Fields: id name standard

```java
package Beyond_Java_8_2.Q1;

record Student (String id, String name, String standard) {}  4 usages

public class Q1 {
    public static void main(String[] args) {
        Student obj = new Student( id: "1",  name: "Akash", standard: "10th");
        Student obj2 = new Student( id: "2",  name: "Sahil", standard: "10th");
        System.out.println(obj);
        System.out.println(obj2);
    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251
Student[id=1, name=Akash, standard=10th]
Student[id=2, name=Sahil, standard=10th]

Process finished with exit code 0
```

Q2)Make sure that no null values should be used for initialization.

```java
package Beyond_Java_8_2.Q1;

import java.util.Objects;

record Student (String id, String name, String standard) {  4 usages
    public Student(String id, String name, String standard) {  2 usages
        this.id = Objects.requireNonNull(id, message: "id should not be null");
        this.name= Objects.requireNonNull(name, message: "Name cannot be Null");
        this.standard = Objects.requireNonNull(standard, message: "Standard cannot be Null");
    }
}

public class Q1 {
    public static void main(String[] args) {
        Student obj = new Student( id: "1",  name: "Akash", standard: "10th");

        try{
            Student obj2 = new Student( id: "2",  name: null, standard: "10th");
        }catch (Exception e){
            System.out.println("Error : "+e.getMessage());
        }


        System.out.println(obj);
//        System.out.println(obj2);
    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-
Error : Name cannot be Null
Student[id=1, name=Akash, standard=10th]

Process finished with exit code 0
```

Q3)Use equal and hashCode methods with Student records

```java
package Beyond_Java_8_2.Q3;


import java.util.Objects;

record Student (String id, String name, String standard) {  4 usages
    public Student(String id, String name, String standard) {  2 usages
        this.id = Objects.requireNonNull(id,  message: "id should not be null");
        this.name = Objects.requireNonNull(name,  message: "Name cannot be Null");
        this.standard = Objects.requireNonNull(standard,  message: "Standard cannot be Null");
    }
}
public class Q3 {
    public static void main(String[] args) {
        Student obj = new Student( id: "1",  name: "Akash", standard: "10th");
        Student obj2 = new Student( id: "2",  name: "Sahil", standard: "10th");
        System.out.println(obj);
        System.out.println(obj2);

        System.out.println("Equals : "+obj.equals(obj2));
        System.out.println("HashCode : "+obj.hashCode());
    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.260
Student[id=1, name=Akash, standard=10th]
Student[id=2, name=Sahil, standard=10th]
Equals : false
HashCode : 1964271704

Process finished with exit code 0
```

Q4)Use a Sealed class Class concept to create a class hierarchy

```java
package Beyond_Java_8_2.Q4;

sealed class A permits B,C,D{  4 usages  4 inheritors
    void show(){  4 usages  4 overrides
        System.out.println("A");
    }
}

final class B extends A{  2 usages
    @Override  4 usages
    void show(){
        System.out.println("B");
    }
}

sealed class C extends A permits E{  3 usages  1 inheritor
    @Override  4 usages  1 override
    void show(){
        System.out.println("C");
    }
}

non-sealed class D extends A{  2 usages
    @Override  4 usages
    void show(){
        System.out.println("D");
    }
}
```

```
        System.out.println("D");
    }
}
final class E extends C{  2 usages
    @Override  4 usages
    void show(){
        System.out.println("E");
    }
}

public class Q4 {
    public static void main(String[] args) {
        A obj;

        obj = new B();
        obj.show(); // Output: B

        obj = new C();
        obj.show(); // Output: C

        obj = new D();
        obj.show(); // Output: D

        obj = new E();
        obj.show(); // Output: E
    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/
B
C
D
E

Process finished with exit code 0
```

Q5)Mark Child classes as final, sealed, and non sealed and observe their behavior

```
/*
Sealed class A ---
                permits 3 classes ---
                                    B , C , D
                                    now B , C ,D can be final sealed non-sealed

                // B is Final hence cannot be extended by other classes;
                // C is sealed and permits E
                // D is non-sealed , hence can be accessed by other classes(extended by other classes)


                                   // E is Final class which extends C


                 // It is mandatory to extend the class which permits to extend - A should be extended by all B  C  D
                 //

*/
```

```
package Beyond_Java_8_2.Q5;

sealed class A permits B, C, D {  4 usages  4 inheritors
    void show(){  4 usages  4 overrides
        System.out.println("A");
    }
}

final class B extends A {  2 usages
    @Override  4 usages
    void show(){
        System.out.println("B");
    }
}

sealed class C extends A permits E {  3 usages  1 inheritor
    @Override  4 usages  1 override
    void show(){
        System.out.println("C");
    }
}

non-sealed class D extends A {  2 usages
    @Override  4 usages
    void show(){
        System.out.println("D");
    }
}
```

```
final class E extends C {  2 usages
    @Override  4 usages
    void show(){
        System.out.println("E");
    }
}
```

Q6)Demonstrate the use of addFirst(), addLast, removeFirst(), removeLast, getFirst(), getLast(), reversed() in Set and List Sequenced collections

```java
package Beyond_Java_8_2.Q6;

import java.util.*;

public class List {

    public static void list() {
        SequencedCollection<String> list = new LinkedList<>();

        list.addFirst( e: "B");
        list.addLast( e: "C");
        list.addFirst( e: "A");
        list.addLast( e: "D");

        System.out.println("List: " + list);

        System.out.println("First: " + list.getFirst());
        System.out.println("Last: " + list.getLast());

        list.removeFirst(); // removes A
        list.removeLast();  // removes D

        System.out.println("After removeFirst & removeLast: " + list);

        SequencedCollection<String> reversedList = list.reversed();
        System.out.println("Reversed List: " + reversedList);
    }
}
```

```java
import java.util.LinkedHashSet;
import java.util.SequencedSet;

public class Set {  no usages

    public static void set(){  no usages
        SequencedSet<String> set = new LinkedHashSet<>();
        set.add("B");
        set.add("C");
        set.addFirst( e: "A");
        set.addLast( e: "D");

        System.out.println("Set: " + set);

        System.out.println("First: " + set.getFirst());
        System.out.println("Last: " + set.getLast());

        set.removeFirst();
        set.removeLast();

        System.out.println("After removeFirst & removeLast: " + set);

        SequencedSet<String> reversedSet = set.reversed();
        System.out.println("Reversed Set: " + reversedSet);

    }
}
```

```java
package Beyond_Java_8_2.Q6;

import static Beyond_Java_8_2.Q6.List.list;
import static Beyond_Java_8_2.Q6.Set.set;

public class Q6 {
    public static void main(String[] args) {
        list();
        System.out.println("\n\n");
        set();
    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/i
List: [A, B, C, D]
First: A
Last: D
After removeFirst & removeLast: [B, C]
Reversed List: [C, B]



Set: [A, B, C, D]
First: A
Last: D
After removeFirst & removeLast: [B, C]
Reversed Set: [C, B]

Process finished with exit code 0
```

Q7)Demonstrate the use of firstEntry(), lastEntry(), pollFirstEntry(), pollLastEntry(), putFirst(), putLast(), reversed() with SequencedMap.

```java
package Beyond_Java_8_2.Q7;

import java.util.*;

public class Q7 {
    public static void main(String[] args) {
        SequencedMap<String, Integer> map = new LinkedHashMap<>();

        map.put( k: "B",  v: 2);
        map.putLast( k: "C",  v: 3);
        map.putFirst( k: "A",  v: 1);
        map.putLast( k: "D",  v: 4);

        System.out.println("Map: " + map);


        System.out.println("First Entry: " + map.firstEntry());
        System.out.println("Last Entry: " + map.lastEntry());


        map.pollFirstEntry();
        map.pollLastEntry();

        System.out.println("After polling first and last: " + map);


        SequencedMap<String, Integer> reversed = map.reversed();
        System.out.println("Reversed map: " + reversed);

    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/idea_rt.jar=330
Map: {A=1, B=2, C=3, D=4}
First Entry: A=1
Last Entry: D=4
After polling first and last: {B=2, C=3}
Reversed map: {C=3, B=2}

Process finished with exit code 0
```