**Iterations Lab Two**

**Document Id 3**

In this section we worked through some introductory material for iterations with Loops and counting conditions within those loops.

**Questions**

1 ) The difference between Arrays and Lists for iteration Algorithms are

1. Arrays are of fixed length while Lists can can have a dynamic size
2. Arrays can only contain primitive types
3. Lists can contain mixed types
4. Lists are an immutable type
5. None of the above

2 ) The reason the static class method Arrays.asList( ) is dangerous is ..., identify all true answers

1. static code can not be used in instance methods
2. It returns a List of undetermined type
3. It returns a immutable list
4. It returns an array ‘represented’ as a list
5. Arrays.asList( ) is fine to use whenever you want to transform an Array to a List

3) The enhanced for loop is better because..., identify all true answers

1. It's a cleaner and easy to read syntax
2. It explicitly uses the Collection Iterator
3. It implicitly uses the Collection Iterator
4. It implicitly uses the Collection Iterator to avoid UnsupportedOperation Exceptions
5. It implicitly uses the Collection Iterator to avoid Conncurrentmodification Exceptions

4) which expression runs without exceptions

1. **List<Model> resultSet = Arrays.asList(modelArray);**

**resultSet .add(1);**

1. **int[] intArray3 = new int[] {1,2,3};**

**List <Integer> dynamicList3 = Arrays.stream(intArray3)**

**.boxed()**

**.collect(Collectors.toList());**

1. **List <Model> dynamicModelList = new ArrayList<>();**

// Model values added to list correctly

**for(Model i : dynamicModelList) {**

**if(**// a correct conditional expression**) {**

**dynamicList.remove(i); }**

**}**

1. **Set <Dto> dtoSet = new HashSet<>();**

**// Dtovalues added to Set correctly**

**Iterator itr = dtoSet .iterator();**

**while(itr.hasNext()) {**

**if( (**// a correct conditional expression**) {**

**dtoSet .remove();**

**}**

**}**

**e) Map<Integer, String> map = new HashMap<>();**

**Collection<String> values = map.values();**

**Iterator itr = values.iterator();**

**while(itr.hasNext()) {**

**if( (// a correct conditional expression) {**

**values .remove();**

**}**

**}**

**Lab Two Practical Iterator compared to the get(index)**

**Background**

At a job interview you are asked to explain how for high performance code that using the the Java Collection Type LinkedList can be dangerous. To answer this you offer to quickly write some code that illustrates the issues.

Write simple code that shows how for certain operations **(get and next)** as shown below

**LinkedList queue = new LinkedList ();**

**Iterator itr = queue.iterator()**

**// get(index) operation**

**queue.get(i);**

**// Iterator operation**

**itr.next()**

Write simple code that shows that the get operation is an order of magnitude slower than the next operation