OCP QUESTION 11

Given:

```
class Dog {
    String name;
    String breed;
    public Dog (String name, String breed) {
        this.name = name;
        this.breed = breed;
    }
    public String getName() { return name; }
    public String getBreed() { return breed; }
}
```

and the code fragment:

```
List<Dog> kennels = Arrays.asList (
    new Dog ("Oliver", "Collie"),
    new Dog ("Sam", "Beagle"),
    new Dog ("Jack", "Beagle"));
kennels.stream()
// line n1
       .collect(Collectors.toList());
```

Which code fragment, when inserted at line n1, sorts the list of dogs in descending order of breed and then ascending order of name?

A. `.sorted(Comparator.comparing(Dog::getBreed).reversed().thenComparing(Dog::getName))`

B. `.sorted(Comparator.comparing(Dog::getBreed).thenComparing(Dog::getName))`

C. `.map(Dog::getBreed).sorted(Comparator.reverseOrder())`

D. `.map(Dog::getBreed).sorted(Comparator.reverseOrder().map(Dog::getName).reversed())`

OCP QUESTION 24

Given:

```
class MutualFund{
    int capital;
    String name;
    public MutualFund (int capital, String name) {
        this.capital = capital;
        this.name = name;
    }
    public String toString () {
        return capital + ":" + name;
    }
}
```

and this code fragment:

```
Set<MutualFund> funds = new TreeSet<>();
funds.add(new MutualFund (293, "Pimco"));
funds.add(new MutualFund (190, "Vanguard"));
System.out.println(funds);
```

What is the result?

A. 293 Pimco
190 Vanguard
B. 190 Vanguard
293 Pimco
C. A compilation error occurs
D. A ClassCastException is thrown at run time

OCP QUESTION 43

Given:

```
class Band {
    String name, style, country;
    public Band (String name, String style, String country) {
        this.style = style;
        this.name = name;
        this.country = country;
    }
    public String getStyle() {
        return style;
    }
    public String toString() {
        return style + " : " + name + " : " + country;
    }
}
```

and the code fragment:

```
List<Band> bands = Arrays.asList(
    new Band("Yes", "Prog Rock", "UK"),
    new Band("Boney M", "Euro Disco", "Germany"),
    new Band("ELP", "Prog Rock", "UK"));
bands.stream()
     .collect(Collectors.groupingBy(Band::getStyle))
     .forEach( (x, y) -> System.out.println(x) );
```

What is the result?

A. [Euro Disco : Boney M : Germany]
[Prog Rock : Yes : UK, Prog Rock : ELP : UK]

B. Euro Disco
Prog Rock

C. [Prog Rock : Yes : UK, Prog Rock : ELP : UK]
[Euro Disco : Boney M : Germany]

D. A compilation error occurs

OCP QUESTION 45

Given the code fragment:

```
public class MapPetShow {
    public static void main (String [ ] args) {

        Map<Integer, String> sourceMap = new HashMap<>( );
        sourceMap.put(37, "Ms.Piggy ");
        sourceMap.put(8, "Gonzo ");
        sourceMap.put(4, "Rowlf ");
        sourceMap.put(12, "Fozzie ");
        sourceMap.put(82, "Kermit ");

        Map<Integer, String> finalMap = new TreeMap<Integer, String> (
            new Comparator<Integer> ( ) {
                @Override
                public int compare(Integer obj1, Integer obj2) {
                    return obj2.compareTo(obj1);
        }});

        finalMap.putAll(sourceMap);
        for (Map.Entry<Integer, String> entry : finalMap.entrySet()) {
            System.out.print(entry.getValue());
        }
    }
}
```

What is the result?

A. A compilation error occurs
B. Rowlf Gonzo Fozzie Ms.Piggy Kermit
C. Kermit Ms.Piggy Fozzie Gonzo Rowlf
D. Rowlf Ms.Piggy Kermit Fozzie Gonzo

OCP QUESTION 56

Given the definition of the ExamTaker class:

```
public class ExamTaker {
    private String name;
    private Integer score;
    ExamTaker(String name, Integer score) {
        this.name = name;
        this.score = score;
    }
    public String getName() {return name;}
    public Integer getScore() {return score;}
}
```

and code fragment:

```
List<ExamTaker> list = Arrays.asList(
    new ExamTaker("Alice", 98),
    new ExamTaker("Bob", 64),
    new ExamTaker("Chris", 72));
Predicate<ExamTaker> passed = e -> e.getScore() >= 65;          // line n1
list = list.stream().filter(passed).collect(Collectors.toList());
Stream<String> names = list.stream().map(ExamTaker::getName);  // line n2
names.forEach(x -> System.out.print(x + " "));
```

What is the result?

A. Alice Bob Chris
B. Alice Chris
C. A compilation error occurs at line n1
D. A compilation error occurs at line n2

OCP QUESTION 65

Given:

```
class CommCodes {
    public static int checkLength(String str1, String str2) {
        return str2.length() - str1.length();
    }
}
class Test{
    public static void main(String[] args) {
        String[] strArr = new String[]{"Sitrep", "Over", "Out", "Roger"};

        //line n1

        for (String s : strArr) {
            System.out.print(s + " ");
        }
    }
}
```

Which code fragment should be inserted at line n1 to enable the code to print Sitrep Roger Over Out?

A. Arrays.sort(strArr, CommCodes :: checkLength);
B. Arrays.sort(strArr, (new CommCodes()) :: checkLength);
C. Arrays.sort(strArr, (CommCodes :: new).checkLength);
D. Arrays.sort(strArr, CommCodes :: new :: checkLength);

OCP QUESTION 66

Given the code fragment:

```
Map<Integer, String> students = new TreeMap<>();
students.put (17, "Alice");
students.put (12, "Chuck");
students.put (11, "Brian");
students.put (13, "Bob");
System.out.println (students);
```

What is the result?

A. {17 = Alice, 12 = Chuck, 11 = Brian, 13 = Bob}
B. {11 = Brian, 12 = Chuck, 13 = Bob, 17 = Alice}
C. {12 = Chuck, 13 = Bob, 17 = Alice}
D. {17 = Alice, 11 = Brian, 13 = Bob, 12 = Chuck}

OCP QUESTION 82

Given:

```
public class Record<K, V> {
    private K key;
    private V value;
    public Record(K key, V value) {
        this.key = key;
        this.value = value;
    }
    public static <T> Record<T, T> write(T value) {
        return new Record<T, T>(value, value); }
    public K getKey() {return key;}
    public V getVal() {return value;}
}
```

Which option fails?

A. `Record<String, Integer> employee = new Record<String, Integer> ("Smith", 25);`

B. `Record<String, String> quote = Record.<String> write("Knowledge is power.");`

C. `Record<Object, Object> score = new Record<String,Integer> ("Joe Random", 86);`

D. `Record<String, String> animation =new Record<>("Shaun The Sheep","Season 5");`

OCP QUESTION 84

Given the definition of the Coder class:

```
public class Coder {
    enum Gender {
         FEMALE, MALE,
    }
    String name;
    Gender gender;
    public Coder(String name, Gender gender) {
        this.name = name;
        this.gender = gender;
    }
    public String getName() {
        return name;
    }
    public Gender getGender() {
        return gender;
    }
}
```

and the code fragment:

```
List<Coder> list = Arrays.asList(
        new Coder("Alice", Coder.Gender.FEMALE),
        new Coder("Chuck", Coder.Gender.MALE),
        new Coder("Bob", Coder.Gender.MALE));
Map<Coder.Gender, List<String>> classification = list.stream().
        collect(Collectors.groupingBy(Coder::getGender,
                Collectors.mapping(Coder::getName, Collectors.toList())));
System.out.println(classification);
```

What is the output?

A. {MALE = [Chuck, Bob], FEMALE = [Alice]}
B. {FEMALE = [Alice], MALE = [Chuck, Bob]}
C. {MALE = [Bob, Chuck], FEMALE = [Alice]}
D. {MALE = [Bob], MALE = [Chuck], FEMALE = [Alice]}

OCP QUESTION 93

Given the code fragments:

```
public class Stock implements Comparator<Stock> {
    String ticker;
    double price;
    public Stock() {
    }
    public Stock(String ticker, double price) {
        this.ticker = ticker;
        this.price = price;
    }
    public int compare(Stock s1, Stock s2) {
        return s1.ticker.compareTo(s2.ticker);
    }
    public String toString() {
        return ticker + ":" + price;
    }
}
```

and

```
List<Stock> portfolio = Arrays.asList(
        new Stock("MSFT", 74),
        new Stock("GOOGL", 952));
Collections.sort(portfolio, new Stock());
System.out.print(portfolio);
```

What is the result?

A. [GOOGL:952.0, MSFT:74.0]

B. [MSFT:74.0, GOOGL:952.0]

C. A compilation error occurs because the Stock class does not override the abstract method `compareTo()`.

D. An Exception is thrown at run time.

OCP QUESTION 94

Given:

```
public interface Printable<Integer> {
    public default void printOnDemand(Integer copies) {
        System.out.println("Printing on demand.");
    }
    public void offsetPrinting(Integer copies);
}
```

Which statement is true?

A. Printable can be used as below:

```
Printable<Integer> publisher
    = x -> System.out.println("Printing " + x + " copies.");
publisher.offsetPrinting(100000);
publisher.printOnDemand(2);
```

B. Printable can be used as below:

```
Printable<Integer> publisher = x -> x + 10;
publisher.offsetPrinting(100000);
publisher.printOnDemand(2);
```

C. Printable can be used as below:

```
Printable publisher = (Integer x) -> System.out.println(x);
publisher.offsetPrinting(100000);
Printable.printOnDemand(2);
```

D. Printable cannot be used in a lambda expression.