Question 1

Which interface in the java.util.function package can return a primitive type?

Options:

A- ToDoubleFunction

B- Supplier

C- BiFunction

D- LongConsumer

Answer:

A

Question 2

Question Type: **MultipleChoice**

Which two statements correctly describe capabilities of interfaces and abstract classes? (Choose two.)

A- Interfaces cannot have protected methods but abstract classes can.

B- Both interfaces and abstract classes can have final methods.

C- Interfaces cannot have instance fields but abstract classes can.

D- Interfaces cannot have static methods but abstract classes can.

E- Interfaces cannot have methods with bodies but abstract classes can.

A, C

Question 3

Question Type: MultipleChoice

Which three annotation uses are valid? (Choose three.)

Options:

A- Function<String, String> func = (@NonNull x) > x.toUpperCase();

B- var v = "Hello" + (@Interned) "World"

C- Function<String, String> func = (var @NonNull x) > x.toUpperCase();

D- Function<String, String> func = (@NonNull var x) > x.toUpperCase();

E- var myString = (@NonNull String) str;

F- var obj = new @Interned MyObject();

Answer:

A, C, F

Question 4

Question Type: MultipleChoice

Consider this method declaration:

```
void setSessionUser(Connection conn, String user) throws SQLException {
    Statement stmt = conn.createStatement();
    String sql = <EXPRESSION>;
    stmt .execute();
}
```

A) "SET SESSION AUTHORIZATION " + user

B) "SET SESSION AUTHORIZATION " + stmt.enquoteIdentifier(user)

Is A or B the correct replacement for and why?

Options:

A- A, because it sends exactly the value of user provided by the calling code.

B- B, because enquoting values provided by the calling code prevents SQL injection.

C- A and B are functionally equivalent.

D- A, because it is unnecessary to enclose identifiers in quotes.

E- B, because all values provided by the calling code should be enquoted.

Answer:

A

Question 5

Which two safely validate inputs? (Choose two.)

Options:

A- Delegate numeric range checking of values to the database.

B- Accept only valid characters and input values.

C- Use trusted domain-specific libraries to validate inputs.

D- Assume inputs have already been validated.

E- Modify the input values, as needed, to pass validation.

Answer:

A, B

Question 6

A bookstore's sales are represented by a list of Sale objects populated with the name of the customer and the books they purchased.

```
public class Sale {

private String customer;

private List items;

// constructor, setters and getters not shown

}

public class Book {

private String name;

private double price;

// constructor, setters and getters not shown

}
```

Given a list of Sale objects, tList, which code fragment creates a list of total sales for each customer in ascending order?

```
A. List<String> totalByUser = tList.stream()
        .collect(flatMapping(t -> t.getItems().stream(),
                groupingBy(Sale::getCustomer,
                summingDouble(Book::getPrice))))
        .entrySet().stream()
        .sorted(Comparator.comparing(Entry::getValue))
        .collect(mapping(e -> e.getKey() + ":" + e.getValue(),toList()));

B. List<String> totalByUser = tList.stream()
        .collect(groupingBy(Sale::getCustomer,
                flatMapping(t -> t.getItems().stream(),
                summingDouble(Book::getPrice))))
        .sorted(Comparator. comparing (Entry::getValue))
        .collect(mapping(e -> e.getKey() + ":" + e.getValue(),toList()));

C. List<String> totalByUser = tList.stream()
        .collect(groupingBy(Sale::getCustomer,
                flatMapping(t -> t.getItems().stream(),
                summingDouble(Book::getPrice))))
        .entrySet().stream()
        .sorted(Comparator.comparing(Entry::getValue))
        .collect(mapping(e -> e.getKey() + ":" + e.getValue(),toList()));

D. List<String> totalByUser = tList.stream()
        .collect(flatMapping(t -> t.getItems().stream(),
                groupingBy(Sale::getCustomer,
                summingDouble(Book::getPrice))))
        .sorted(Comparator.comparing (Entry::getValue))
        .collect(mapping(e -> e.getKey() + ":" + e.getValue(),toList()));
```

Options:

A- Option A

B- Option B

C- Option C

D- Option D

Answer:

C

Question 7

Question Type: MultipleChoice

Given:

```
public class Confidential implements Serializable{
    private String data;

    public Confidential(String data) {
        this.data = data;
    }
}
```

Which two are secure serialization of these objects? (Choose two.)

A- Define the serialPersistentFields array field.

B- Declare fields transient.

C- Implement only readResolve to replace the instance with a serial proxy and not writeReplace.

D- Make the class abstract.

E- Implement only writeReplace to replace the instance with a serial proxy and not readResolve.

Answer:

A, C

Question 8

Question Type: **MultipleChoice**

Which two statements set the default locale used for formatting numbers, currency, and percentages? (Choose two.)

Options:

A- Locale.setDefault(Locale.Category.FORMAT, "zh-CN");

B- Locale.setDefault(Locale.Category.FORMAT, Locale.CANADA_FRENCH);

C- Locale.setDefault(Locale.SIMPLIFIED_CHINESE);

D- Locale.setDefault("en_CA");

E- Locale.setDefault("es", Locale.US);

Answer:

B, D