Unit 1:

1)Which is an invalid variable name?

$_exceptionValue

MyNewValue

object

4ScoreAnd8Years

_SystemValue

2) Java uses ____ to structure the code, and ____ for individual instances.

classes, references

classes, objects

classes, attributes

objects, classes

3) Source code is saved in ____ files, and code is compiled into ____ files.

.java, .bytes

.text, .java

.java, .exe

.java, .class

4) Which statements are true about Java? (Choose two)

Java source code must be compiled into classes for each platform.

Java is a specialized programming language for use in browsers.

Java source code is plain text.

Java is an object-oriented programming language.

Unit -2

1) Which is not a Java primitive type?

float

String

boolean

long

short

2) Examine the following code:

```
int x = 1, y = 1, z = 0;
if (x == y | x < ++y) {
       z = x+y;
}
else{
       z = 1;
}
System.out.println(z);
```

What would be printed?

2

3

1

Because: x=1 y=1

Then ++y y=1 become y=2

Now z=x+y is z=1+2

Them is answer is 3…..

3) Why would you use the ternary operator ?: instead of writing an if/else construct?

If you run out of room

When the boolean expression is not to be evaluated

None of the options listed

If you only need to assign a value based on a condition

To change the value returned

4) What is the correct evaluation of this expression: 8*8/2+2-3*2?

30

28

26

10

1. Explation: Multiply 8 by 8: `64`
2. Divide the result by 2: `64 / 2 = 32`
3. Add 2 to the result: `32 + 2 = 34`
4. Subtract 3 multiplied by 2: `34 - 3 * 2 = 34 - 6 = 28`

So, the correct evaluation of the expression is `28`. Therefore, the answer is:28..

5) Which is the correct way to declare and initialize a variable?

age = 42;

char midInit = "D";

boolean x = 4>5;

int age;

unit 3:

1)What must you use to apply object-oriented capabilities to primitives?

Wrapper classes

Reflection Classes

Class casting

StringBuilder

Class object

2) Which statements are true about StringBuilder? (Choose three)

StringBuilder objects are mutable.

StringBuilder objects automatically expand capacity as needed.

You can instantiate a StringBuilder with a predefined content or capacity.

StringBuilder objects are thread-safe.

3) Which statement is NOT true about Formatter classes?

They are used to parse text.

They work with messages, numbers, date, and time values.

They work with only objects, but not primitives.

They are used to format text.

4) Which statement is not true about Strings in Java?

String is a class.

Strings allow you to append text.

Always creates a single copy of each string literal.

Strings represent a sequence of characters.

Unit-4

1) Which statements are true about a Class diagram? (Choose two)

It documents access modifiers.

It illustrates method implementations.

It shows how one class invokes another.

It shows how one class is related to another.

2) Which UML diagrams are used to model program flow of control? (Choose two)

Activity diagram

Sequence diagram

State Transition diagram

Class diagram

3) Which statement is true?

A class must contain import statements for classes in other packages that it uses.

Object must be explicitly referenced.

Class must explicitly define the package of which it is a member.

An object must be an instance of a specific class.

4) Which statements are true about methods? (Choose two)

Objects may contain method definitions to implement behaviors of their classes.

Methods may describe a comma-separated list of parameters.

A void method may not contain a return statement.

Method body is enclosed with "{" and "}" symbols.

Methods must contain a return statement.

5) What is an object reference?

A typed variable that points to an object's location in memory

Constructed based on the structure of the object

Explicitly garbage collected by the programmer

A pointer to specific fields that make make up an object

Unit – 5

1) Which two statements are true about enum?

It cannot be used in switch constructs.

It cannot be used as a variable value.

It provides a fixed set of instances of a specific type.

Enum values are instances of this enum type.

Enum values are implicitly protected.

Enum cannot be used as a variable type

2) Information contained within an object should normally be "hidden" inside it. What is this called?

Shadowing

Encapsulation

Abstraction

Inheritance

Polymorphism

3) Which statements are true about a constructor method? (Choose two)

It must not have an access modifier.

It must be named after the class.

It is a special method that initializes the object.

It must be void.

It must have one or more parameters.

4) Which statements are true? (Choose two)

Variables are allocated in a stack.

Each thread has its own stack.

Objects are allocated in the heap.

Primitives cannot be placed into a heap.

Enum cannot be used in switch constructs.

5) Which are rules for method overloading? (Choose two)

Must be in different classes

Must have the same return type

Must have a different number or types of parameters, or both

Must not have identical names

Must not be void

Must have different parameter names

Unit 6:

1) Assume that Food is a subclass of Product and examine the following code:

```
public void order(Product product) {
      if ( /* condition */ ) {
            LocalDate bestBefore =
((Food)product).getBestBefore();
      }
}
```

Which IF condition should be used to verify the object type before casting the reference?

(Product instanceof food)

(product.equals(Food))

(product.instanceOf(Food))

(product instanceof Food)

(food instanceof Product)

(Food instanceof product)

2) Which are public methods of the Object class? (Choose three)

wait()

toString()

hashCode()

clone()

3) Which statements are true? (Choose two)

The Object class is the ultimate parent of any other class in Java.

The Object class defines common, generic operations that all other Java classes inherit and reuse.

The "extends Object" clause is implied even when an explicit extends clause is present.

Class Object constructor may not be explicitly invoked.

4) Which is NOT a rule of reference type casting?

Casting is possible between objects of sibling types.

Casting is required when invoking a polymorphic operation.

Casting is required to assign a parent to child reference type.

No casting is required to assign a child to parent reference type.

5) Which statements are true about the abstract keyword? (Choose two)

Abstract classes make polymorphism impossible.

Concrete subclasses must override all abstract methods of their abstract parent.

The Abstract class can contain only final variables and methods.

The Abstract class must be extended by one or more concrete subclasses.

The Abstract class cannot be directly instantiated.

Unit 7 :

1) What is an interface that defines a single abstract operation called?

Inherited

Polymorphic

Abstract

Functional

Unary

2) Interfaces can contain concrete methods, but only if they are ___. (Choose three)

final

static

overloaded

private

overriden

default

3) Which action do Generics allow?

Hardcoding the exact type as part of the class design

Resolving specific type during compilation

Wrapping values within the class by using the type Object

Assigning any type to a variable or parameter whose type is Object

4) Which statements are true? (Choose three)

A class can choose if and when to override default methods.

A default method can be defined only in an interface.

A class can implement as many interfaces as required.

Interfaces help solve the multiple inheritance problem.

An abstract class may not override default methods.

5) Which statements are true? (Choose two)

Both interface and class references can be cast to either type.

An interface reference cannot be cast to another interface type.

An interface reference provides access to all public methods of the class.

An interface reference cannot be cast to the class type.

A class reference cannot be cast to the interface type.

An interface works with the instanceof operator.

Unit : 8

1) Examine the following code:

```
String[] names = {"Mary","Jane","Ann","Tom"};
Arrays.sort(names);
int x = Arrays.binarySearch(names,"Ann");
System.out.println(x);
```

What is the value of x?

4

1

0

2

3

2) When processing an array in a loop, use array ____ to determine the boundary for the termination condition:

length

size()

length()

capacity

size

3) Examine the following code:

```
public class Compare implements Comparator<String>{
  public int compare(String s1, String s2) {
    return s2.length() - s1.length();
  }
}

String[] names = {"Mary","Jane","Elizabeth","Jo"};
Arrays.sort(names, new Compare());
for (String name: names) {
  System.out.println(name);
}
```

What will be the output?

Mary Jane Elizabeth Jo

Elizabeth Jane Mary Jo

Jo Jane Mary Elizabeth

Elizabeth Mary Jane Jo

Jo Mary Jane Elizabeth

4) When breaking out of loops and skipping loop cycles, ____ skips the current loop and ____ terminates the current loop.

continue, return

continue, break

break, continue

return, break

5) Which statements are true about an array? (Choose four)

It is a fixed-length collection of elements of the same type indexed by int.

An array of primitive values is filled with 0 values (false values if it is of boolean type).

After an array is created, its length cannot be changed.

Is an object itself

You must "extract" the array element from the array before you operate on it.

Unit 9 :

1) Examine the following code:

```
String[] arr = {"Tea","Cake"};
List<String> texts = Arrays.asList(arr);
```

Which statements are true? (Choose two)

You can add a new String to texts.

You can replace Tea with Coffee in arr.

You can replace Tea with Coffee in texts.

You can add a new String to arr.

2) Which statement is true about Set?

Null element cannot be added to a HashSet.

A Set cannot be populated with a List, because a List allows duplicate elements.

Add and Remove methods will return false values when attempting to add a duplicate or remove an absent element.

Add and Remove methods throw an exception when attempting to add a duplicate or remove an absent element.

3) Examine the following code:

```
Map<Integer, String> items = new HashMap<>();
items.put(Integer.valueOf(1),"Tea");
items.put(Integer.valueOf(2),"Cake");
```

Which statements are true? (Choose two)

items.put(Integer.valueOf(3),"Cake"); will create an additional element with a "Cake" value.

items.put(Integer.valueOf(2),"Cake"); will throw an exception.

items.put(Integer.valueOf(3),"Tea"); will replace the integer key value for the "Tea" element.

items.put(Integer.valueOf(1),"Coffee"); will replace "Tea" with "Coffee."

4) Which statements are NOT true about the Java Collection API? (Choose two)

All collections dynamically expand as appropriate to accommodate new elements.

All collections provide thread-safe operations.

All collections allow programs to store groups of objects in memory.

All collections are implemented internally by using arrays.

5) Which statements are true about Deque? (Choose three)

Null values are not allowed.

peekFirst(T) and peekLast(T) returns and removes the element at the head or tail of the Deque.

If the deque is empty, pool and peek operations throw an exception.

offerFirst(T) and offerLast(T) insert elements at the head and the tail of the deque.

Implementations of ArrayDeque will auto-expand.

Unit – 10:

1) With what is a member inner class associated?

Static context of the containing class

Context of a specific method

Instance context of the containing class

Inline implementation of an interface

2) Which statement is true about lambda expressions?

It is an inline implementation of a functional interface.

A lambda expression's parameter types can always be inferred.

A lambda expression algorithm always contains a single line of code.

It can always be used instead of any anonymous inner class.

3) Why would you define classes inside other classes?

To constrain the use of the class and hide it from the containing class

To simplify logic and constrain context of use

To encapsulate logic and constrain context of use

To reuse logic and allow the class to be part of the containing class

4) Which statements are true about an anonymous inner class? (Choose two)

It can access outer variables, but only if they are final or effectively final.

It is an implementation of an interface or extension of a class.

It can be an implementation of a functional interface.

It can access only private variables in the containing class.

Unit – 11:

1) What would you use while processing a stream to replace every element in a stream with a different value?

Supplier

Function

Predicate

Consumer

2) What kind of interface does the filter method in a stream accept?

Supplier

Map

Consumer

Predicate

3) Which statement is NOT true about Java Streams?

Stream pipeline traversal uses method chaining.

Pipeline traversal occurs starting with the first stream method encountered.

Stream processing can be sequential.

After an element is processed, it is no longer available from the stream.

4) Which statement is true about intermediate stream operations?

They traverse the stream and end the stream processing.

They do not have primitive variants.

At least one is always required in a stream.

They perform an action and produce another stream.

Unit – 12

1) The most commonly used logging method takes at least two parameters. Which parameters are they?

Level and message

Localization and message

Level and localization

Precision and message

2) Which statement is true about the Java Logging API?

You can apply different log levels to different log targets.

A given log message can be written to only one log target.

When log level is set to INFO priority, SEVERE messages are ignored.

A common practice is to use "Logger" as the logger name.

3) What happens when an exception is raised?

Control is passed to the nearest similar exception handler.

Normal program flow is terminated.

Normal program flow is suspended until the exception is handled.

Some form of message is included in the exception's constructor.

4) Which statement is true about checked exceptions?

Their presence is evidence of a bug in your code.

They inherit directly from java.util.Error.

They may be caught and need not be explicitly propagated.

The must be caught or must be explicitly propagated.

5) What are the common uses of custom exceptions? (Choose two)

Eliminating the need to catch checked exceptions

Providing constructors that utilize superclass constructor abilities

Providing an error message

Avoiding the need to include exception information in a method signature

Unit – 13

1) Which statement is NOT true about serialization?

Deserialization is a process of reading objects from the stream.

Data is serialized in binary form.

Serialization is a suitable solution for long-term data storage.

Serialization is a process of writing objects from memory into a stream.

2) Which statement is NOT true about java.nio.file classes?

Class Path represents files and folders.

Class Path allows you to discover file system roots.

Class FileSystem describes available file systems and their properties.

Class Path allows you to set file system object properties.

3) Which is NOT a basis for categorizing Java I/O Streams?

The type of data that stream can carry, for example: text or binary

Whether it is serial or parallel

Type of source or destination to which this stream is connected

Direction of the stream: input or output

4) Assume that a file system has an empty folder /users/joe and the following code:

```
Path backup = Path.of("/users/joe/backup/docs");
Files.createDirectory(backup);
```

What is the result?

It will not do anything.

It will throw an exception.

It will create backup and docs folders.

It will create the backup folder only.

Unit – 14

1) What is the state of two threads waiting on each other in a synchronized block called?

Deadlock

ReadWriteLock

Starvation

Livelock

2) Which are valid thread state transitions? (Choose two)

WAITING to TERMINATED

TERMINATED to RUNNABLE

NEW to RUNNABLE

RUNNABLE to BLOCKED

BLOCKED to TERMINATED

3) When you catch the InterruptedException, into which state does it put the thread?

NEW

RUNNABLE

INTERRUPTED

TERMINATED

4) Which statement is true about threads?

The order in which threads terminate is indeterminate.

A high-priority thread will complete its task earlier than a low-priority thread.

Two concurrent threads can enter the same synchronized block simultaneously.

You can always force a thread to terminate.

5) Which are valid mechanisms to implement threads? (Choose two)

Instantiate a Runnable object.

Create a class to extend Runnable.

Schedule a thread to run.

Create a class that implements Thread.

Unit -15

1) Which statements are true? (Choose three)

Classes of a given package must all be within the same module.

Explicitly exported packages are visible to all other modules

Packages provide the logical grouping of classes.

All public classes are hidden access unless the containing package is explicitly exported.

All classes of a given package become visible to other modules when exported.

2) Which statements are true about services? (Choose two)

A service provider references a service module with the "requires" directive.

A service consumer references a service module with the "requires" directive.

A service provider references a service module with the "uses" directive.

A service consumer references a service provider module with the "requires" directive.

3) Which statement is true about modules?

The "requires transitive" directive enables class reflection in the corresponding module.

The "opens" directive enables class reflection in the corresponding module.

The "exports" directive describes a permission to use modules.

The "requires static" directive includes static classes from the corresponding module.

4) When legacy JAR files are placed into the module path, what are they treated as?

Headless

Transitive

Migrated

Automatic

Partice question…..

1) Given:

```
public class Store {
	public String desc;
	@CustomAnnotation(owner = "Sam")
	public Store(String desc) {
		this.desc = desc;
	}
}
```

What is a possible implementation for this annotation?

```
@Inherited
public @interface CustomAnnotation {
	String owner();
	int count();
}
```

```
@Target(value = { FIELD, CONSTRUCTOR })
public @interface CustomAnnotation {
	String owner();
	int count() default 100;
}
```

```
public @interface CustomAnnotation {
	String owner = "Bob";
	int count = 100;
}
```

```
@Target(value = { METHOD })
public @interface CustomAnnotation {
	String owner();
	int count() default 100;
}
```

2) given:

```
int x = 0;
while(x < 100){
	if(x%3 == 0){
		x += 3;
	}
	if(x%2 == 0){
		x += 2;
		continue;
	}
	x++;
}
```

What is the value of x after the fourth iteration?

- 10
- 24
- 11
- **12**

3) Given:

```
public static Function<String,String> swap = s -> {
	if(s.equals("Australia"))
		return "New Zealand";
	else
		return s;
	};
```

And given:

```
Set<String> islandNations = Set.of("Australia", "Japan",
"Taiwan", "Cyprus", "Cuba");
islandNations = islandNations.stream()
	.map(swap)
	.map(n -> n.substring(0, 1))
	.collect(Collectors.toSet());
for(String s : islandNations){
	System.out.print(s);
}
```

What is the output?

- TCAJC
- TCAJ
- UnsupportedOperationException
- TCNJ

CTJN

TJNC

TCNJC

4) Given:

```
Path someFile = Path.of("/","users","joe","docs","some.txt");
Path justSomeFile = someFile.getFileName();
Path docsFolder = someFile.getParent();
Path currentFolder = docsFolder.relativize(someFile);
System.out.println(currentFolder);
```

What is the output of this code?

some.txt

joe/some.txt

users/joe

/users/joe/docs

5) Given:

```
class Main {
	public static void main(String[] args) {
		Scanner myObj = new Scanner(System.in);
		String input;
		input = myObj.nextLine();
		System.out.println("The input is: " + input);
	}
}
```

What is the result when you enter null while executing this program?

null

The input is: null

RuntimeException

NullPointerException

6) Given:

```
public class Square {
	private double length;
	public Square(double length) {
		this.length = length;
```

```
       }
         public double findSurfaceArea(){
                 return length*length;
          }
}
```

And given:

```
public class Cube extends Square {
       public Cube(double length) {
              super(length);
       }
       public double findSurfaceArea() {
              return super.findSurfaceArea() * 6;
       }
}
```

And given:

```
public static void main(String[] args){
       Square shape = new Cube(1);
       System.out.println(shape.findSurfaceArea());
}
```

What is the result?

1

Does not compile

36

6

7) Given:

```
var cities = List.of(new City("Berlin", 3_520_000),
       new City("Hamburg", 1_790_000),
       new City("Munich", 1_450_000),
       new City("Cologne", 1_060_000),
       new City("Frankfurt", 730_000));
cities.stream()
       .filter(city -> city.getPopulation() < 1_000_000)
       .findFirst()
       .orElse(new City("Not Found", 0));
```

The List interface

The City Class

The Stream interface

The Optional class

8) Given the code fragment:

```
List<String> list = ...;
list.forEach( x -> { System.out.println(x); } );
```

What is the type of x?

- Stream
- ArrayList
- **String**
- List
- Consumer

9) Given:

```
for(int i=0; i<5; i++) {
      System.out.println(++i);
}
```

Which is an equivalent statement?

```
int i=0;
 while(i<5){
    i++;
    System.out.println(++i);
 }
```

```
for(int i=0; i<5; ){
    System.out.println(++i++);
 }
```

```
int i=0;
 while(i<5){
    System.out.println(++i);
    ++i;
 }
```

```
for(int i=0; ; i++){
    if(i<5)
       continue;
    System.out.println(++i);
 }
```

10) Given:

```
module codeHaus {
	exports pkgA;
	exports pkgB to modOne;
	exports pkgC to modOne, modTwo;
}
module modOne {
	//Insert code here
}
```

What can be inserted in modOne module-info to access pkgC public constructs in module codeHaus?

requires pkgA, pkgB, pkgC;

requires modTwo;

requires codeHaus;

requires pkgB, pkgC;

11) Given:

```
public class Square {
	public int calcArea(int x) {
		return (x * x);
	}
}
```

And given:

```
public class Cube extends Square {
	public int calcArea(int x) {
		return super.calcArea(x * 6);
	}
}
```

And given:

```
public static void main(String[] args) {
	Cube shape = new Square();
	System.out.println(shape.calcArea(1));
}
```

What is the result of trying to run this main method?

Compilation fails

Both 1 and 6 print

6

1

12) Which cause a compilation error? (Choose two)

```
public abstract class testClass {
	double taxRate = 0.05;
	abstract void increaseTax(){
		taxRate = taxRate + 0.01;
	}
}
public abstract class testClass {
	double taxRate = 0.05;
	public void increaseTax() {
		taxRate = taxRate + 0.01;
	}
}
public interface TestInterface {
	static double taxRate = 0.05;
	abstract void increaseTax();
}
public interface TestInterface {
	double taxRate = 0.05;
	void increaseTax();
}
public interface TestInterface {
	double taxRate = 0.05;
	default void increaseTax() {
		taxRate = taxRate + 0.01;
	}
}
```

13) Given:

```
try {
	InputStream in = new FileInputStream("missingfile.txt");
	in.read();
	in.close();
}
catch(/*line n1*/) {
	System.out.println(e.getClass().getName());
}
catch(/*line n2*/) {
	System.out.println(e.getClass().getName());
}
```

Which two changes, made together, allow this code to compile? (Choose two)

Insert at line n1: FileNotFoundException e

Insert at line n2: IOException e

Insert at line n1: Exception e

Insert at line n1: IOException e

Insert at line n2: FileNotFoundException e

14) Which statements are valid? (Choose two)

```
BiPredicate<Integer, Integer> test = (x, y) -> x == y;
Supplier<String> hello = () -> "Hello";
Function<String> hello = () -> "Hello";
BiPredicate<Integer, Integer> add = (x, y) -> x + y;
DoubleConsumer cube = x -> x*x*x;
```

15) Which statement causes a compilation error?

```
DoubleFunction<Double> area = r -> Math.PI*r*r;
DoublePredicate<Integer, Integer> test = (x, y) -> x == y;
BiFunction<Integer, Integer, Integer> divide = (x, y) ->
x/y;
IntPredicate test = x -> x == 10;
```

16) Given:

```
module A {
      exports pkgA to B;
      requires D;
}
module B {
      requires transitive A;
}
module C {
      exports pkgC to D;
      requires B;
}
module D {
      exports pkgD;
      requires C;
}
```

Why does this code fail to compile?

Module B does not export anything.

Module A is not exported to Module D.

Module A is not exported to Module C.

There is a cyclic dependency.

17) Given:

```
public double findSurfaceArea(double length){
	return length*length;
}
```

Which one is a valid overloaded method?

```
public void findSurfaceArea(double length){
	System.out.println(length*length);
}
```

```
public double findSurfaceArea2(double length){
	return length*length;
}
```

```
public double findSurfaceArea(int length){
	return length*length;
}
```

```
private double findSurfaceArea(double length){
	return length*length;
}
```

18) Given:

```
public enum Planet {
	MERCURY(0.39), VENUS(), EARTH;
	private double distanceFromSunAU = 1.0;
	private Planet(double distance){
		distanceFromSunAU = distance;
	}
	private Planet(){
		distanceFromSunAU = 5.0;
	}
	public double getDistace(){
		return distanceFromSunAU;
	}
}
```

And:

```
public static void main(String[] args) {
	for(Planet p : Planet.values()){
		System.out.println(p.getDistace());
	}
}
```

What is the output when running main method?

0.39 5.0 5.0

0.39 1.0 5.0

0.39 1.0 1.0

This does not compile.

0.39 5.0 1.0

19) Given:

```
public String pickName(){
    List names = List.of("Barclay", "Barry", "Bert", "Bort");
    /*line 1*/
    return names.stream()
        .filter(n -> n.contains("Bart"))
        .findAny()
        /*line 2*/;
}
```

Which changes, made independently, compile and do not throw any exceptions? (Choose two)

Insert `names.add("Bart");` at line 1.

Insert `.orElse("Not Found")` at line 2.

Insert `.get()` at line 2.

Change the method return type from String to Optional.

Change the method return type from String to Stream.

20) Which statements are valid? (Choose two)

```
Consumer<String> con3 = final x -> System.out.print(x);
```

```
BiConsumer<String, String> biCon1 = (final var x, final var
y) -> System.out.println(x+y);
```

```
BiConsumer<String, String> biCon2 = (final var x, y) ->
System.out.println(x+y);
```

```
Consumer<String> con2 = final var x -> System.out.print(x);
```

```
BiConsumer<String, String> biCon3 = (final var x, var y) ->
System.out.println(x+y);
```

```
Consumer<String> con1 = final String x ->
System.out.print(x);
```

21) Given:

```
int x = 0;
while(x < 100){
	if(x%2 == 1){
		x++;
	}
	else if(x%2 == 0){
		x += 3;
	}
}
```

What is the value of x after the fourth iteration?

19

8

15

11

22) Given:

```
public interface InterfaceOne {
	int field = 99;
	default void method() {
		System.out.print(field);
	}
}
public interface InterfaceTwo {
	int field = 1983;
	private void method() {
		System.out.print(field);
	}
}
public class TheConcrete implements InterfaceOne, InterfaceTwo
{
	public static void main(String[] args) {
		TheConcrete theInstance = new TheConcrete();
		theInstance.method();
	}
}
```

What is the result of running this code?

Compiler error. There is a multiple inheritance of state caused by field.

Prints: 1983

Prints: 99

Compiler error. TheConcrete must implement method.

23) Given the code fragment:

```
public class MyInClass {
    public static void main(String[] args) {
        String s = "";
        try (BufferedReader in = new BufferedReader(new
InputStreamReader(System.in))) {
            System.out.print("Type e to exit: ");
            s = in.readLine();
            while (s != null) {
                System.out.println("Output: " + s.trim());
                if (s.equals("e")) {
                    System.exit(0);
                }
                System.out.print("Type e to exit: ");
                s = in.readLine();
            }
        } catch (IOException e) {
            System.out.println("Exception: " + e);
        }
    }
}
```

What is the result when you enter null while executing this program?

The program exits.

The program prints Output: null

An IOException is thrown.

The program prints Output:

24) Given:

```
int month = 11;
switch(month) {
      case 1: case 3: case 5: case 7:
      case 8: case 10: case 12:
            String message = "31 days. ";
            System.out.print(message);
      case 4: case 6: case 9: case 11:
            String message = "30 days. ";
            System.out.print(message);
      case 2:
```

```
            String message = "28 days. ";
            System.out.print(message);
}
```

What is the output of this code?

30 days.

31 days. 30 days.

Does not print because of compilation error

30 days. 28 days.

25) Given this code in the file Circle.java:

```
public class Test {
      public static void main(String args[]) {
            double cir = Circle.findCircumference(7.5);
            System.out.print("Circumference of circle=" +
cir);
      }
}
public class Circle {
      public static double findCircumference(int radius) {
            return 2 * Math.PI * radius;
      }
}
```

What can be done to allow this code to run?

Change the findCircumference method return type.

Change the findCircumference method parameter type.

Call java Circle instead.

Separate the classes into two .java files.

Call javac Circle first.

Create an instance of Circle before calling findCircumference.

26) Given:

```
10  public class Pet {
11      private var name;
12      private var breed = "German Shepherd";
```

```
13      private static final List<Pet> petList = new
ArrayList<>();
14      public Pet(String name, var breed){
15          this.name = name;
16          this.breed = breed;
17          petList.add(this);
18      }
19      public static void printPets(){
20          for(var pet : petList){
21              System.out.println(pet.name +"", ""
+pet.breed);
22          }
23      }
24  }
```

Which use of var is valid?

- Line 11
- Line 12
- **Line 20**
- Line 14

27) Given:

```
public interface Black {
      default void getColor(){
            System.out.print("Black");
      }
}
public interface Red extends Black {
      default void getColor(){
            System.out.print("Red");
      }
}
public interface Gold {
      static void getColor(){
            System.out.print("Gold");
      }
}
public class Colors implements Black, Red, Gold {
      public static void main(String[] args) {
            Colors colors = new Colors();
            colors.getColor();
      }
}
```

What is the result of running this code?

Compiler error, error: types Black and Red are incompatible;

Prints: Gold

Prints: Black

Prints: Red

28) Given:

```
// line n1
public interface Pet {
      public abstract int breed();
      // line n2
      public default Pet callPet(){
            return this;
      }
      // line n3
      void speak();
}
```

What is a valid annotation?

@FunctionalInterface at line n1

@interface at line n1

@Inherited at line n1

@Deprecated(forRemoval=true) at line n3

@Override at line n2

@Target(METHOD) at line n3

29) Given:

```
1. public class Transaction {
2.           private static int counter = 0;
3.           private int transactionID;
4.           private String customerName;
5.
6.           public void executeTransaction(String
customerName) {
7.
8.                 transactionID = counter;
9.                 counter++;
10.                  this.customerName = customerName;
11.
12.   }
```

```
13. }
```

How do you make this code thread safe?

Change line 1 to `public class synchronized Transaction{`

Change line 2 to `private synchronized static int counter = 0;`

Change line 7 to `synchronized(this){` and line 11 to `}`

Change line 6 to `public synchronized (this) executeTransaction(String customerName){`

30) Given:

```
11 public interface FuncInteface {
12          int sum = 0;
13          final static int divisor = 3;
14          double quotient();
15          default void product() { System.out.println(3); }
16          abstract int remainder(int x, int y);
17          static int result() { return 0; }
18          private int count() { return sum; }
19 }
```

What should you do to make this interface a functional interface? (Choose two)

Remove line 12

Remove line 15

Remove line 16

Remove line 14

Remove line 14 and 16

Remove line 17

Remove line 13 and 17

31)

Given:

```
public class Test {
      public Test() {
            System.out.print("Message 1" +", ");
      }
```

```
	public Test(String message) {
		System.out.print(message +", ");
		display();
	}
	public void display() {
		System.out.print("Message 2" +", ");
	}
	public void display(Object message) {
		System.out.print("Object" +", ");
	}
	public void display(String message) {
		System.out.print(message +", ");
	}
}
```

And the code fragment:

```
Test test = new Test();
test.display("Message X");
```

What is the output?

Message X, Message 2, Message X,

Message 1, Message X,

Message X, Message 2,

Message 1, Object,

32) Given:

```
public class MyBSClass {
	public static void main(String[] args) {
		byte[] b = new byte[256];
		try (FileInputStream fis = new
FileInputStream(args[0]);
				FileOutputStream fos = new
FileOutputStream(args[1])) {
			int count = 0;
			int read = 0;
			while ((read = fis.read(b)) != -1) {
				fos.write(b);
				count += read;
			}
			System.out.println("Output: " + count);
		} catch (Exception e) {
			System.out.println(e);
		}
	}
}
```

...executed using this command:

```
java MyBSClass file0.txt file1.txt
```

What does this code do?

Copy anything beyond the initial 256 bytes of data from file0.txt into file1.txt.

Print the size of file0.txt.

Copy only the first 256 bytes of data from file1.txt into file0.txt.

Print 256.

Print the difference in file sizes between file0.txt and file1.txt.

33) Given:

```
public interface TestInterface {
    public abstract double calculateTax();
}
```

Which classes compile? (Choose two)

```
public class testClass implements TestInterface {
      public double calculateTax(double tax){...}
}
```

```
public class testClass {
      public double calculateTax(){...}
}
```

```
public class testClass implements TestInterface {
      public abstract double calculateTax();
}
```

```
public abstract class testClass implements TestInterface {
      public double calculateTax(double tax){...}
}
```

```
public abstract class testClass implements TestInterface {
      public abstract void calculateTax(){...}
}
```

34) Given:

```
public class MyChannelClass {
      public static void main(String[] args) {
            try (FileChannel fcIn = new
FileInputStream(args[0]).getChannel();
            FileChannel fcOut = new
FileOutputStream(args[1]).getChannel()) {
                  ByteBuffer buff = ByteBuffer.allocate((int)
fcIn.size());
```

```
                    System.out.println("Output: " +
buff.remaining());
              } catch (Exception e) {
                    System.out.println(e);
              }
       }
}
```

On executing this command:

```
java MyChannelClass file0.txt file1.txt
```

What does this code do?

Prints the size of file0.text minus the size of file1.txt.

Prints the size of file0.txt.

Writes the file size of file0.txt into file1.txt.

Prints the size of file1.txt.

Copies the content of file0.txt into file1.txt.

35) Given:

```
public interface TheInterface {
       int field = 99;
       abstract void method();
}
public abstract class TheAbstract {
       int field = 1983;
       abstract void method();
}
public class TheConcrete extends TheAbstract implements
TheInterface {
       void method() {
              System.out.println("balloons");
       }
}
```

Why does compilation fail?

Interfaces cannot have instance fields

TheConcrete class using the wrong access modifier for method()

Multiple inheritance of state from field variable

Multiple inheritance of behavior from method()

36) Given:

```
module A {
```

```
      exports pkgA;
      requires B;
      //line n1
}
```
And given:
```
open module B {
      exports pkgB;
      //line n2
}
```
Which is a valid statement?

opens pkgA; at line n1

requires B to pkgA; at line n1

requires A; at line n2

requires pkgB; at line n1

opens pkgB to pkgA; at line n2

37) Examine these module JAR files and their module-info java files:

```
//order.jar:
module order {
      requires product;
      exports com.oracle.order;
}
//product.jar:
module product {
      exports com.oracle.product;
}
```
Which is the only possible result of executing the command `jdeps -s order.jar product.jar`?

```
order -> product
```

```
java.base -> product
java.base -> order
product -> order
```

```
order -> java.base
order -> product
product -> java.base
```

```
product -> order
```

```
order -> product
product -> order
```

38) Which is a valid syntax?

```
Consumer<Integer> lambda = (x) -> {
	x++;
	System.out.print(x);
};
BiConsumer<String, String> lambda = (String x,y) ->
System.out.print(x+y);
String city = "Munich";
BiFunction<Integer, Integer, String> lambda =
String::substring;
Function<Double, Double> lambda = r -> {
	double circumference = 2*Math.PI*r;
};
```

39) Given:

```
public class Planet {
	private BigDecimal milesFromSun;
	public Planet(String milesFromSun){
		this.milesFromSun = new BigDecimal(milesFromSun);
	}
	public BigDecimal getMilesFromSun(){
		return milesFromSun;
	}
	public static BigDecimal milesToKilometers(BigDecimal
miles){
		BigDecimal scale = new BigDecimal("1.60934");
		return miles.multiply(scale);
	}
}
```

And given:

```
public static void main(String[] args) {
	Planet earth = new Planet("93000000");
	//line n1
}
```

Which statement is valid at line n1?

```
Function<Planet, BigDecimal> lambda =
earth::milesToKilometers;

Function<Planet, BigDecimal> lambda =
Planet::milesToKilometers;

Function<BigDecimal, BigDecimal> lambda =
earth.getMilesFromSun()::milesToKilometers;

Supplier<BigDecimal> lambda =
()::milesToKilometers(earth.getMilesFromSun());
```

```
Supplier<BigDecimal> lambda =
earth.getMilesFromSun()::milesToKilometers;
```

```
Function<BigDecimal, BigDecimal> lambda =
Planet::milesToKilometers;
```

39) Which statements cause a compiler error when written inside the main method? (Choose two)

```
var double = 10.0f;
```

```
var list1 = new ArrayList<>();
list1.add("Otto");
var char1 = list1.get(0).charAt(0);
```

```
var list2 = new ArrayList<String>();
list2.add("Hans");
var char2 = list2.get(0).charAt(0);
```

```
var name = "Gertrud";
var char3 = name.charAt(0);
```

```
var var = 10;
```