30 JAVA.

1.

What will be the output of this program?

```
class Color {
int red, green, blue;
void Color() {
red = 10;
green = 10;
blue = 10;
}
void printColor() {
System.out.println("red: " + red + " green: " + green + " blue: " +
blue);
}
public static void main(String [] args) {
Color color = new Color();
color.printColor();
}
}
```

A. Compiler error: no constructor provided for the class
B. Compiles fine, and when run, it prints the following: red: 0 green: 0 blue: 0
C. Compiles fine, and when run, it prints the following: red: 10 green: 10 blue: 10
D. Compiles fine, and when run, crashes by throwing `NullPointerException`

Answers: B

2.

Consider the following program and predict the behavior of this program:

```
class Base {
public void print() {
System.out.println("Base:print");
}
}
abstract class Test extends Base { //#1
public static void main(String[] args) {
Base obj = new Base();
obj.print(); //#2
}
}
```

A. Compiler error “an abstract class cannot extend from a concrete class” at statement marked with comment #1
B. Compiler error “cannot resolve call to print method” at statement marked with comment #2
C. the program prints the following: `Base:print`
D. the program will throw a runtime exception of `AbstractClassInstantiationException`

Answers: C

3.

```
class Base {}
class DeriOne extends Base {}
class DeriTwo extends Base {}
class ArrayStore {
public static void main(String []args) {
Base [] baseArr = new DeriOne[3];
baseArr[0] = new DeriOne();
baseArr[2] = new DeriTwo();
System.out.println(baseArr.length);
}
}
```

Which one of the following options correctly describes the behavior of this program?
a. this program prints the following: 3

B. this program prints the following: 2
C. this program throws an `ArrayStoreException`
D. this program throws an `ArrayIndexOutOfBoundsException`
Answers: C

4.

```
class Color {
int red, green, blue;
Color() {
Color(10, 10, 10);
}
Color(int r, int g, int b) {
red = r;
green = g;
blue = b;
}
void printColor() {
System.out.println("red: " + red + " green: " + green + " blue: " +
blue);
}
public static void main(String [] args) {
Color color = new Color();
color.printColor();
}
}
```

A. Compiler error: cannot find symbol
B. Compiles without errors, and when run, it prints: red: 0 green: 0 blue: 0
C. Compiles without errors, and when run, it prints: red: 10 green: 10 blue: 10
D. Compiles without errors, and when run, crashes by throwing
`NullPointerException`
Answers: A

5

Choose the correct option based on this code segment:

```
class Rectangle { }
class ColoredRectangle extends Rectangle { }
class RoundedRectangle extends Rectangle { }
class ColoredRoundedRectangle extends ColoredRectangle, RoundedRectangle { }
```

Choose an appropriate option:
A. Compiler error: '{' expected cannot extend two classes
B. Compiles fine, and when run, crashes with the exception
`MultipleClassInheritanceException`
C. Compiler error: class definition cannot be empty
D. Compiles fine, and when run, crashes with the exception
`EmptyClassDefinitionError`
Answers: A

6

Consider the following program and determine the output:

```
class Test {
public void print(Integer i) {
System.out.println("Integer");
}
public void print(int i) {
System.out.println("int");
}
public void print(long i) {
System.out.println("long");
}
public static void main(String args[]) {
Test test = new Test();
test.print(10);
```

```
}
}
```

A. the program results in a compiler error (“ambiguous overload”)
B. `long`
C. `Integer`
D. `int`

Answers: D

7

Consider the following code and choose the right option for the word <access-modifier>:

```
// Shape.java
public class Shape {
protected void display() {
System.out.println("Display-base");
}
}
// Circle.java
public class Circle extends Shape {
<access-modifier> void display(){
System.out.println("Display-derived");
}
}
```

A. Only protected can be used
B. public and protected both can be used
C. public, protected, and private can be used
D. Only public can be used

Answers: B

8

Which of the following method(s) from `Object` class can be overridden? (select all that apply.)

A. `finalize()` method
B. `clone()` method
C. `getClass()` method
D. `notify()` method
E. `E.wait()` method

Answers: A

9

Choose the correct option based on the following program:

```
class Color {
int red, green, blue;
Color() {
this(10, 10, 10);
}
Color(int r, int g, int b) {
red = r;
green = g;
blue = b;
}
public String toString() {
return "The color is: " + red + green + blue;
}

public static void main(String [] args) {
System.out.println(new Color());
}
}
```

A. Compiler error: incompatible types
B. Compiles fine, and when run, it prints the following: the color is: 30
C. Compiles fine, and when run, it prints the following: the color is: 101010

D. Compiles fine, and when run, it prints the following: the color is:
red green blue

Answers: C

10.

Choose the best option based on the following program:

```
class Color {
int red, green, blue;
Color() {
this(10, 10, 10);
}
Color(int r, int g, int b) {
red = r;
green = g;
blue = b;
}
String toString() {
return "The color is: " + " red = " + red + " green = " + green + "
blue = " + blue;
}
public static void main(String [] args) {
// implicitly invoke toString method
System.out.println(new Color());
}
}
```

A. Compiler error: attempting to assign weaker access privileges; `toString` was public in `Object`
B. Compiles fine, and when run, it prints the following: the color is: red = 10 green = 10 blue = 10
C. Compiles fine, and when run, it prints the following: the color is: red = 0 green = 0 blue = 0
D. Compiles fine, and when run, it throws `ClassCastException`

Answers: A

11

Choose the correct option based on this code segment:

```
LocalDate babyDOB = LocalDate.of(2015, Month.FEBRUARY, 20);
LocalDate now = LocalDate.of(2016, Month.APRIL, 10);
System.out.println(Period.between(now, babyDOB).getYears()); // PERIOD_CALC
```

a. the code segment results in a compiler error in the line marked with the comment `PERIOD_CALC`
B. the code segment throws a `DateTimeException`
C. the code segment prints: `1`
D. the code segment prints: `-1`

Answers: D

12

Which one of the following classes is best suited for storing timestamp values of application events in a file?

a. `java.time.ZoneId` class
B. `java.time.ZoneOffset` class
C. `java.time.Instant` class
D. `java.time.Duration` class
e. `java.time.Period` class

Answers: C

13

Given this code segment

```
ZoneId zoneId = ZoneId.of("Asia/Singapore");
ZonedDateTime zonedDateTime =
ZonedDateTime.of(LocalDateTime.now(), zoneId);
System.out.println(zonedDateTime.getOffset());
```

assume that the time-offset value for the Asia/Singapore time zone from

UTC/Greenwich is +08:00. Choose the correct option.
a. this code segment results in throwing `DateTimeException`
B. this code segment results in throwing `UnsupportedTemporalTypeException`
C. the code segment prints: `Asia/Singapore`
D. the code segment prints: `+08:00`
e. this code segment prints: `+08:00 [Asia/Singapore]`
Answers: D

14

Choose the correct option based on this code segment:

```
DateTimeFormatter dateFormat = DateTimeFormatter.ISO_DATE; // DEF
LocalDate dateOfBirth = LocalDate.of(2015, Month.FEBRUARY, 31);
System.out.println(dateFormat.format(dateOfBirth)); // USE
```

a.the program gives a compiler error in the line marked with the comment `DEF`
B. the program gives a compiler error in the line marked with the comment `USE`
C. the code segment prints: `2015-02-31`
D. the code segment prints: `2015-02-03`
e. this code segment throws `java.time.DateTimeException` with the message `"Invalid date 'FEBRUARY 31'"`
Answers: E

15

Consider this code segment:

```
DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("EEEE", Locale.US);
System.out.println(formatter.format(LocalDateTime.now()));
```

Which of the following outputs matches the string pattern `"EEEE"` given in this code segment?
a. F
B. Friday
C. sept
D. september

Answers: B

16

Choose the correct option based on this program:

```
import java.util.function.ObjIntConsumer;
class ConsumerUse {
public static void main(String []args) {
ObjIntConsumer<String> charAt = (str, i) -> str.charAt(i); // #1
System. out.println(charAt.accept("java", 2)); // #2
}
}
```

a. this program results in a compiler error for the line marked with comment #1
B. this program results in a compiler error for the line marked with comment #2
C. this program prints: a
d. this program prints: v
e. this program prints: 2

Answers: D

17

Choose the correct option based on this program:

```
import java.util.function.Predicate;
public class PredicateTest {
public static void main(String []args) {
Predicate<String> notNull =
((Predicate<String>)(arg -> arg == null)).negate(); // #1
System.out.println(notNull.test(null));
}
}
```

a. this program results in a compiler error in line marked with the comment #1
B. this program prints: true
C. this program prints: false
d. this program crashes by throwing `NullPointerException`
Answers: C

18

Choose the correct option based on this program:

```
import java.util.function.Function;
public class AndThen {
public static void main(String []args) {
Function<Integer, Integer> negate = (i -> -i), square = (i -> i * i),
negateSquare = negate.compose(square);
System. out.println(negateSquare.apply(10));
}
}
```

a. this program results in a compiler error
B. this program prints: -100
C. this program prints: 100
d. this program prints: -10
e. this program prints: 10
Answers: B

19.

Which one of the following functional interfaces can you assign the method reference `Integer::parseInt`? note that the static method `parseInt()` in `Integer` class takes a `String` and returns an `int`, as in: `int parseInt(String s)`
a. Bipredicate<string, Integer>
B. Function<Integer, string>
C. Function<string, Integer>
d. predicate<string>
e. Consumer<Integer, string>
F. Consumer<string, Integer>
Answers: C

20.

Choose the correct option based on this program:

```
import java.util.function.BiFunction;
public class StringCompare {
public static void main(String args[]){
BiFunction<String, String, Boolean> compareString = (x, y) ->
x.equals(y);
System.out.println(compareString.apply("Java8","Java8")); // #1
}
}
```

a. this program results in a compiler error in line marked with #1
B. this program prints: true
C. this program prints: false
d. this program prints: `(x, y) -> x.equals(y)`
e. this program prints: `("Java8", "Java8") -> "Java8".equals("Java8")`

Answers: B

21

Which one of the following interfaces declares a single abstract method named `iterator()`? (Note: Implementing this interface allows an object to be the target of the for-each statement.)

a) `Iterable<T>`
b) `Iterator<T>`
c) `Enumeration<E>`
d) `ForEach<T>`

Answers:A

22

Choose the correct option based on this program:

```
import java.util.stream.Stream;
public class Reduce {
public static void main(String []args) {
Stream<String> words = Stream.of("one", "two", "three");
int len = words.mapToInt(String::length).reduce(0, (len1, len2) ->
len1 + len2);
System.out.println(len);
}
}
```

a) this program does not compile and results in compiler error(s)
b) this program prints: onetwothree
c) this program prints: 11
d) this program throws an `IllegalArgumentException`

Answers: C

23

Which one of the following options is best suited for generating random numbers in a multi-threaded application?

a) Using `java.lang.Math.random()`
b) Using `java.util.concurrent.ThreadLocalRandom`
c) Using `java.util.RandomAccess`
d) Using `java.lang.ThreadLocal<T>`

Answers: B

24

Given this code segment:

```
DateTimeFormatter fromDateFormat = DateTimeFormatter.ofPattern("MM/dd/yyyy");
// PARSE_DATE
DateTimeFormatter toDateFormat = DateTimeFormatter.ofPattern("dd/MMM/YY");
System.out.println(firstOct2015.format(toDateFormat));
```

Which one of the following statements when replaced with the comment PARSE_DATE will result in the code to print "`10/Jan/15`"?

a) `DateTimeFormatter firstOct2015 = DateTimeFormatter.`
`parse("01/10/2015", fromDateFormat);`
b) `LocalTime firstOct2015 = LocalTime.parse("01/10/2015",`
`fromDateFormat);`
c) `Period firstOct2015 = Period.parse("01/10/2015", fromDateFormat);`
d) `LocalDate firstOct2015 = LocalDate.parse("01/10/2015",`
`fromDateFormat);`

Answers: D

25

Consider the following program:

```
import java.util.*;
class ListFromVarargs {
public static <T> List<T> asList1(T... elements) {
ArrayList<T> temp = new ArrayList<>();
for(T element : elements) {
```

```
temp.add(element);
}
return temp;
}
public static <T> List<?> asList2(T... elements) {
ArrayList<?> temp = new ArrayList<>();
for(T element : elements) {
temp.add(element);
}
return temp;
}
public static <T> List<?> asList3(T... elements) {
ArrayList<T> temp = new ArrayList<>();
for(T element : elements) {
temp.add(element);
}
return temp;
}
public static <T> List<?> asList4(T... elements) {
List<T> temp = new ArrayList<T>();
for(T element : elements) {
temp.add(element);
}
return temp;
}
}
```

Which of the `asList` definitions in this program will result in a compiler error?

a) the definition of `asList1` will result in a compiler error
b) the definition of `asList2` will result in a compiler error
c) the definition of `asList3` will result in a compiler error
d) the definition of `asList4` will result in a compiler error
e) None of the definitions (`asList1, asList2, asList3, asList4`) will result in a compiler error

Answers: B

26

Given this code segment:

```
IntFunction<UnaryOperator<Integer>> func = i -> j -> i * j;
// LINE
System.out.println(apply);
```

Which one of these statements when replaced by the comment marked with LINE will print 200?

a) `Integer apply = func.apply(10).apply(20);`
b) `Integer apply = func.apply(10, 20);`
c) `Integer apply = func(10 , 20);`
d) `Integer apply = func(10, 20).apply();`

Answers:A

27

Given this code segment:

```
List<Map<List<Integer>, List<String>>> list = new ArrayList<>(); // ADD_MAP
Map<List<Integer>, List<String>> map = new HashMap<>();
list.add(null); // ADD_NULL
list.add(map);
list.add(new HashMap<List<Integer>, List<String>>()); // ADD_HASHMAP
list.forEach(e -> System. out.print(e + " ")); // ITERATE
```

Which one of the following options is correct?

a) this program will result in a compiler error in line marked with comment `ADD_MAP`
b) this program will result in a compiler error in line marked with comment `ADD_HASHMAP`
c) this program will result in a compiler error in line marked with comment `ITERATE`

d) When run, this program will crash, throwing a `NullPointerException` in line marked with comment `ADD_NULL`
e) When run, this program will print the following: `null {} {}`

Answers:E

28

Given this code snippet:

```
LocalDate dateOfBirth = LocalDate.of(1988, Month.NOVEMBER, 4);
MonthDay monthDay =
MonthDay.of(dateOfBirth.getMonth(), dateOfBirth.getDayOfMonth());
boolean ifTodayBirthday =
monthDay.equals(MonthDay.from(LocalDate.now())); // COMPARE
System.out.println(ifTodayBirthday ? "Happy birthday!" : "Yet another day!");
```

Assume that today's date is 4th November 2015. Choose the correct answer based on this code segment.

a) this code will result in a compiler error in the line marked with the comment `COMPARE`
b) When executed, this code will throw `DateTimeException`
c) this code will print: `Happy birthday!`
d) this code will print: `Yet another day!`

Answers:C

29

Consider the following program:

```
class Base<T> { }
class Derived<T> { }
class Test {
public static void main(String []args) {
// Stmt #1
}
}
```

Which statements can be replaced with `// Stmt#1` and the program remains compilable (choose two):

a) `Base<Number> b = new Base<Number>();`
b) `Base<Number> b = new Derived<Number>();`
c) `Base<Number> b = new Derived<Integer>();`
d) `Derived<Number> b = new Derived<Integer>();`
e) `Base<Integer> b = new Derived<Integer>();`
f) `Derived<Integer> b = new Derived<Integer>();`

Answers:A,F

30

Which of the following classes in the `java.util.concurrent.atomic` package inherit from `java.lang.Number`? (Select all that apply.)

a) `AtomicBoolean`
b) `AtomicInteger`
`AtomicLong`
d) `AtomicFloat`
e) `AtomicDouble`

Answers:B,C

30 SPRING.

Question 11

Select one or many correct answers about Spring bean life cycle.

1. The method annotated with @PostConstruct is called after bean instantiation and before properties setting of the bean
2. The method @PreDestroy of a prototype bean is called when the bean is garbage collected
3. The init() method declared in the init-method attribute of a bean is called before the afterPropertiesSet callback method of the InitializingBean interface

4. The method annotated with @PostConstruct is called before the afterPropertiesSet callback method of the InitializingBean interface

Answers: 4

Question 12

Given the following configuration class, what are the correct affirmations? Select one or more answers.

```
public class ApplicationConfig {
private DataSource dataSource;
@Autowired
public ApplicationConfig(DataSource dataSource) {
this. dataSource = dataSource;
}
@Bean(name="clientRepository")
ClientRepository jpaClientRepository() {
return new JpaClientRepository();
}
}
```

1. @Configuration annotation is missing
2. Default or no-arg constructor is missing
3. @Bean name is ambiguous
4. @Bean scope is prototype

Answers: 1 and 2

Question 13

What are the features of the XML <context:namespace? Select one or many answers.

1. @Transactional annotation scanning
2. @Aspect annotation detection enabling
3. @Autowired annotation enabling
4. @Component annotation scanning

Answers: 3 and 4

Question 14

Select one or more correct statements about developing integration test with Spring support.

1. A new Spring context is created for each test class
2. To get a reference on the bean you want to test, you have to call the getBean() method of the Spring context
3. Spring context configuration could be inherited from the super class
4. The Spring context configuration file has to be provided to the @ContextConfiguration annotation

Answers: 3

Question 15

What are the main advantage(s) for using Spring when writing integration tests?

1. Reuse Spring configuration files of the application
2. Create mock or stub
3 . Be able to use the rollback after the test pattern
4. Use dependency injection

Answers: 1,3 and 4

Question 16

What are the main advantage(s) for using Spring when writing unit tests?

1. Reuse Spring configuration files of the application
2. Use dependency injection
3. Provide some mocks for servlet classes

4. All of the above
5. None of the above

Answers: 3

Question17

What is right about the Spring test module?

1. It provides an abstraction layer for the main open source mock frameworks
2. Provides the @Mock annotation
3. It dynamically generates mock objects
4. All of the above
5. None of the above

Answers: 5

Question18

Select correct statement(s) about transactional support of the Spring test module.

1. Transaction manager could be set within the @TransactionConfiguration annotation
2. Method annotated with @Before is executed outside of the test's transaction
3. Spring test may rollback the transaction of a service configured with the REQUIRES_NEW propagation
4. The transaction of a method annotated with the @Rollback annotation with its default values is rolled back after the method has completed

Answers: 1 and 4

Question19

Considering 2 classes AccountServiceImpl and ClientServiceImpl. Any of these 2 classes inherits from each other. What is the result of the following pointcut expression?

execution(* *..AccountServiceImpl.update(..))
&& execution(* *..ClientServiceImpl.update(..))

1. Matches public update methods of the 2 classes, whatever the arguments
2. Matches any update methods of the 2 classes, whatever the arguments and method visibility
3. Matches any update methods of the 2 classes, with one more arguments and whatever method visibility
4. No joint point is defined

Answers: 4

Question20

Using the Spring AOP framework, what is the visibility of the method matches by the following join point?

```
@Pointcut("execution(* *(..))")
private void anyOperation() {};
```

1. All methods, whereas their visibility
2. All methods, except private method
3. Protected and public methods
4. Public methods

Answers: 4

Question21

What are the 2 correct statements about AOP proxy?

1. AOP proxies are created by Spring in order to implement the aspect contracts
2. AOP proxies are always created with a JDK dynamic proxy
3. Only classes that implements a least one interface could be proxied
4. All methods could be proxied
5. Proxies are created by a BeanPostProcessor

Answers: 1 and 5

Question22

What is an after throwing advice? Select a unique answer.

1. Advice that could throw an exception
2. Advice to be executed if a method exits by throwing an exception
3. Advice that executes before a join point
4. Spring does not provide this type of advice

Answers: 2

Question 23

What is an after returning advice? Select a unique answer.

1. Advice to be executed regardless of the means by which a join point exits
2. Advice that surrounds a method invocation and can perform custom behavior before and after the method invocation
3. Advice to be executed before method invocation
4. Advice to be executed after a join point completes without throwing an exception

Answers: 4

Question 24

What is an advice? Select a unique answer.

1. An action taken by an aspect at a particular join point
2. A point during the execution of a program
3. An aspect and a pointcut
4. A predicate that matches join points

Answers: 1

Question 25

What is a pointcut? Select the single answer.

1. Code to execute at a join point
2. An expression to identify joinpoints
3. An advice and a jointpoint
4. None of the above

Answers: 2

Question 34

What is/are incorrect statements about XML declaration of the transaction manager bean? Select one or more answers.

1. The tx namespace provides JTA transaction manager declaration shortcut syntax
2 . Id of the bean has to be transactionManager
3. Depending the application persistence technology, the HibernateTransactionManager or the DataSourceTransactionManager could be used as bean class
4. Default transaction timeout could be given

Answers: 2

Question 35

Assuming @Transactional annotation support is enabled and the transferMoney method is called through a Spring AOP proxy, what is the behavior of the following code sample?

```
@Transactional(propagation=Propagation. R E Q U I R E D)
public void transferMoney(Account src, Account target, double amount) {
add(src, - amount);
add(src, amount);
}
@Transactional(propagation=Propagation. R E Q U I R E S _ N E W)
public void add(Account account, Double amount) {
// IMPLEMENTATION
}
```

1. The add() method executes code in a new transaction
2. The add() method uses the transaction of the transferMoney() method
3. When calling the add() method, an exception is thrown
4. Other behavior

Answers: 2

Question 36

Does Spring provide programmatic transaction management? Select a unique answer.

1. Yes with the TransactionTemplate class
2. Yes with the TransactionService class
3. Yes using the @Transactional bean post processor
4. No

Answers: 1

Question 37

What is the transaction behavior of the PROPAGATION_REQUIRES_NEW mode? Select a unique answer.

1. If a transaction exists, the current method should run within this transaction. Otherwise, it should start a new transaction and run within its own transaction.
2. If a transaction is in progress, the current method should run within the nested transaction of the existing transaction. Otherwise, a new transaction has to be started and run within its own transaction.
3. The current method must start a new transaction and run within its own transaction. If there is an existing transaction in progress, it is suspended.
4. None of the above

Answers: 3

Question 38

What is the default rollback policy in transaction management?

1. Rollback for any Exception
2. Rollback for RuntimeException
3. Rollback for checked exceptions
4. Always commit

Answers:2

Question 42

How could you secure MVC controller with Spring Security? Select a unique answer.

1. With the @Secured annotation
2. With the @RolesAllowed annotation
3. In a XML security configuration file
4. All of the above
5. None of the above

Answers:4

Question 43

What are the possible mechanisms provided by Spring Security to store user details? Select one or more correct answers.

1. Database
2. JAAS
3. LDAP
4. Properties file

Answers: 1,2,3,4

Question 44

What is right about Spring Security configuration and the security namespace? Select one or more correct answers.

1 . The access attribute of the intercept-url tag support both EL and constants together.
2. The patterns declared into the intercept-url tag are analyzed from up to bottom. Winning is

the first that matches.
3. The patterns declared into the intercept-url tag use by default the java regex syntax.
4. Security rules may apply depending request parameter
Answers: 2

Question 49
What provides Spring Boot?
1. Support for Jetty and Undertow as embedded containers
2. Java code generation
3. Auto-configuration of the Spring Framework and third libraries
4. Convenient dependency descriptors to load transitive dependencies
5. Support both Java-based and YAML for Spring application context configuration
Answers: 1,3,4

Question 50
What is the name of the default environment configuration file of Spring Boot?
1. configuration.spring
2. configuration.yml
3. configuration.xml
4. application.properties
5. application.json
Answers: 4

Question 29
Why is it a best practice to mark transaction as read-only when code does not write anything to the database? Select one or more answers.
1. It is mandatory for using Spring exception translation mechanism
2. May be improve performance when using Hibernate
3. Spring optimizes its transaction interceptor
4. Provides safeguards with Oracle and some other databases
Answers: 2 and 4

Question 30
What data access technology is supported by the Spring framework? Select one or more answers.
1. JDBC
2. NoSQL
3. Hibernate
4. JPA
Answers: 1,3 and 4

Question 31
What is not provided by the JdbcTemplate? Select a unique answer.
1. Data source access
2. Open/close data source connection
3. JDBC exception wrapping into DataAccess Exception
4. JDBC statement execution
Answers: 1

Question 32
Using JdbcTemplate, what is the Spring provided class you will use for result set parsing and merging

rows into a single object? Select a unique answer.
1. RowMapper
2. RowCallbackHandler
3. ResultSetExtractor
4. ResultSetMapper

Answers: 3

Question 33

What configuration is supported by the LocalSessionFactoryBean which supports Hibernate 4 or higher? Select a unique answer.
1. Listing entity classes annotated with @Entity
2 . Scanning a package to detect annotated entity classes (with @Entity)
3. Listing hibernate XML mapping configuration file (.hbm.xml)
4. All above

Answers: 4