Object-oriented programming (OOPs)

For more details on SUN Certifications, visit JavaScjpDumps

Ques: 01 Given:

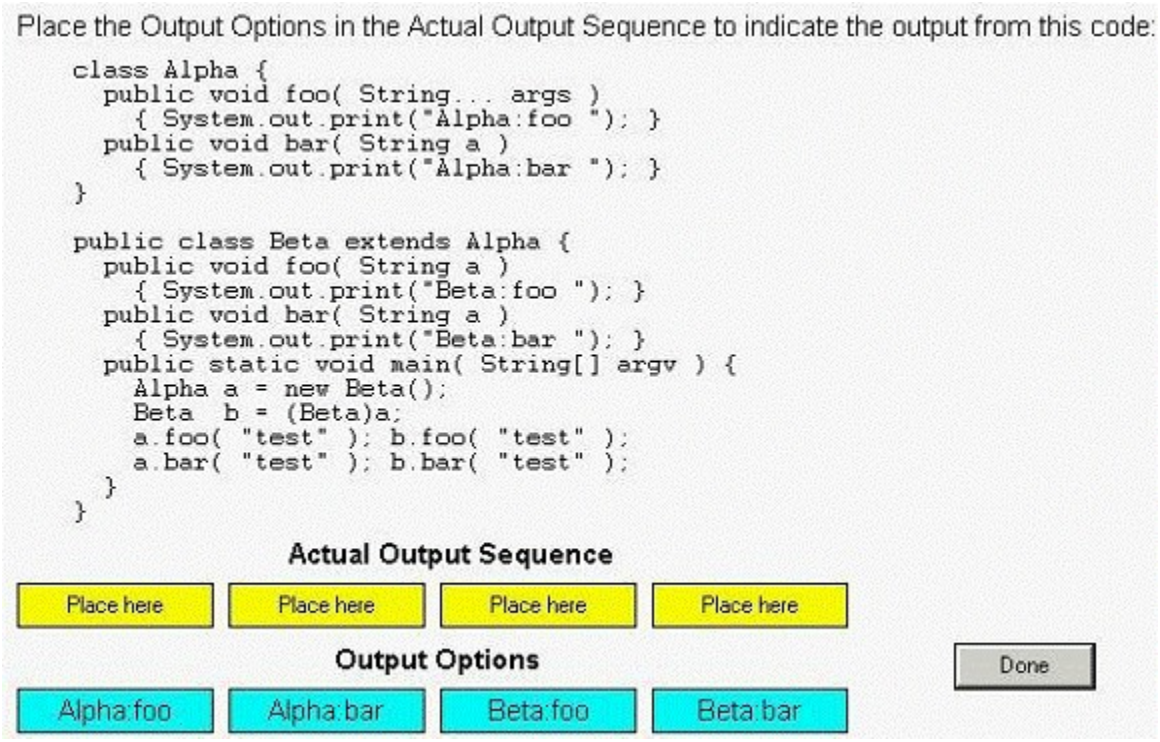
```
public class Plant {
private String name;
public Plant(String name) { this.name = name; }
public String getName() { return name; }
}
1. public class Tree extends Plant {
2. public void growFruit() { }
3. public void dropLeaves() { }
4. }
```

Which statement is true?

A. The code will compile without changes.
B. The code will compile if public Tree() { Plant(); } is added to the Tree class.
C. The code will compile if public Plant() { Tree(); } is added to the Plant class.
D. The code will compile if public Plant() { this("fern"); } is added to the Plant class.
E. The code will compile if public Plant() { Plant("fern"); } is added to the Plant class.

Answer: D

Ques: 02 Click the Task button.



Solution:

Alpha.foo Beta.ba Beta.foo Beta.bar

Ques: 03 Given:

```
11. public class Yikes {
12.
13. public static void go(Long n) {System.out.println("Long ");}
14. public static void go(Short n) {System.out.println("Short ");}
15. public static void go(int n) {System.out.println("int ");}
16. public static void main(String [] args) {
17. short y = 6;
18. long z = 7;
19. go(y);
20. go(z);
21. }
22. }
```

What is the result?

A. int Long
B. Short Long
C. Compilation fails.
D. An exception is thrown at runtime.

Answer: A

Ques: 04 Click the Exhibit button.

Which statement is true about the classes and interfaces in the exhibit?

```
1. public interface A {
2.   public void doSomething(String thing);
3. }

1. public class AImpl implements A {
2.   public void doSomething(String msg) { }
3. }

1. public class B {
2.   public A doit() {
3.     // more code here
4.   }
5.
6.   public String execute() {
7.     // more code here
8.   }
9. }

1. public class C extends B {
2.   public AImpl doit() {
3.     // more code here
4.   }
5.
6.   public Object execute() {
7.     // more code here
8.   }
9. }
```

A. Compilation will succeed for all classes and interfaces.

B. Compilation of class C will fail because of an error in line 2.
C. Compilation of class C will fail because of an error in line 6.
D. Compilation of class AImpl will fail because of an error in line 2.
Answer: C

Ques: 05 Click the Task button.

Given:

```
class A {
  String name = "A";
  String getName() {
    return name;
  }
  String greeting(){
    return "class A";
  }
}
class B extends A {
  String name = "B";
  String greeting() {
    return "class B";
  }
}
public class Client {
  public static void main( String[] args ) {
    A a = new A();
    A b = new B();
    System.out.println(a.greeting() + " has name " + a.getName());
    System.out.println(b.greeting() + " has name " + b.getName());
  }
}
```

Place the names "A" and "B" in the following output.

class [Place here] has name [Place here]

class [Place here] has name [Place here]

Names: [A] [B]

[Done]

Solution:
class A has name A
class B hasname A

Ques: 06 Given:

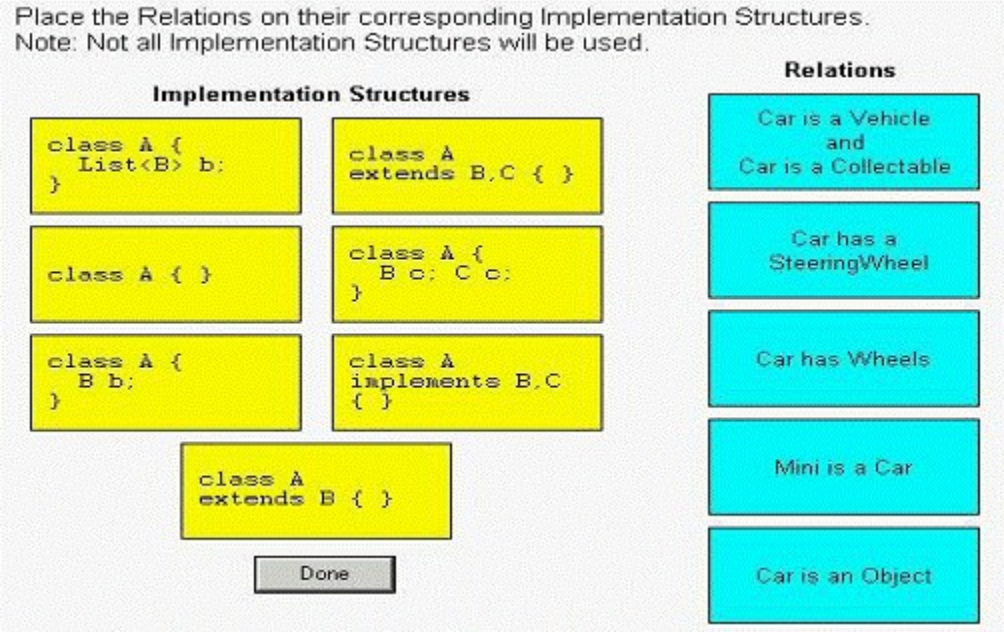
```
10: public class Hello {
11: String title;
12: int value;
13: public Hello() {
14: title += " World";
15: }
16: public Hello(int value) {
17: this.value = value;
18: title = "Hello";
19: Hello();
20: }
21: }
```

and:
30: Hello c = new Hello(5);
31: System.out.println(c.title);
What is the result?
A. Hello
B. Hello World
C. Compilation fails.
D. Hello World 5
E. The code runs with no output.
F. An exception is thrown at runtime.
Answer: C

Ques: 07 Which two statements are true about has-a and is-a relationships? (Choose two.)
A. Inheritance represents an is-a relationship.
B. Inheritance represents a has-a relationship.
C. Interfaces must be used when creating a has-a relationship.
D. Instance variables can be used when creating a has-a relationship.
Answer: A, D

Ques: 08 Given:
1. class Super {
2. private int a;
3. protected Super(int a) { this.a = a; }
4. }
...
11. class Sub extends Super {
12. public Sub(int a) { super(a); }
13. public Sub() { this.a = 5; }
14. }
Which two, independently, will allow Sub to compile? (Choose two.)
A. Change line 2 to:
public int a;
B. Change line 2 to :protected int a;
C. Change line 13 to :public Sub() { this(5); }
D. Change line 13 to :public Sub() { super(5); }
E. Change line 13 to :public Sub() { super(a); }
Answer: C, D

Ques: 09 Click the Task button.



Solution:

Car is a Vehicle and car is a Collectable < ==========> **class A implements B , C { }**

Car has a Steering Wheel < ==========> **class A { B b; }**

car has Wheels < ==========> **class A{ List <B> b; }**

Mini is a Car < ==========> **class A extends B{ }**

Car is an Object < ==========> **class A { }**

Ques: 10 Given:

```
1. interface A { public void aMethod(); }
2. interface B { public void bMethod(); }
3. interface C extends A,B { public void cMethod(); }
4. class D implements B {
5. public void bMethod(){}
6. }
7. class E extends D implements C {
8. public void aMethod(){}
9. public void bMethod(){}
10. public void cMethod(){}
11. }
```

What is the result?

A. Compilation fails because of an error in line 3.
B. Compilation fails because of an error in line 7.
C. Compilation fails because of an error in line 9.
D. If you define D e = new E(), then e.bMethod() invokes the version of bMethod() defined in Line 5.
E. If you define D e = (D)(new E()), then e.bMethod() invokes the version of bMethod() defined in Line 5.
F. If you define D e = (D)(new E()), then e.bMethod() invokes the version of bMethod() defined in Line 9.
Answer: F

Ques: 11 Given:

```
1. class ClassA {
2. public int numberOfInstances;
3. protected ClassA(int numberOfInstances) {
4. this.numberOfInstances = numberOfInstances;
5. }
6. }
7. public class ExtendedA extends ClassA {
8. private ExtendedA(int numberOfInstances) {
9. super(numberOfInstances);
10. }
11. public static void main(String[] args) {
12. ExtendedA ext = new ExtendedA(420);
13. System.out.print(ext.numberOfInstances);
14. }
15. }
```

Which statement is true?
A. 420 is the output.
B. An exception is thrown at runtime.
C. All constructors must be declared public.
D. Constructors CANNOT use the private modifier.
E. Constructors CANNOT use the protected modifier.
Answer: A

Ques: 12 Given:

```
1. public class Base {
2. public static final String FOO = "foo";
3. public static void main(String[] args) {
4. Base b = new Base();
5. Sub s = new Sub();
6. System.out.print(Base.FOO);
```

7. System.out.print(Sub.FOO);
8. System.out.print(b.FOO);
9. System.out.print(s.FOO);
10. System.out.print(((Base)s).FOO);
11. } }
12. class Sub extends Base {public static final String FOO="bar";}
What is the result?
A. foofoofoofoofoo
B. foobarfoobarbar
C. foobarfoofoofoo
D. foobarfoobarfoo
E. barbarbarbarbar
F. foofoofoobarbar
G. foofoofoobarfoo
Answer: D

Ques: 13 Which two statements are true? (Choose two.)
A. An encapsulated, public class promotes re-use.
B. Classes that share the same interface are always tightly encapsulated.
C. An encapsulated class allows subclasses to overload methods, but does NOT allow overriding methods.
D. An encapsulated class allows a programmer to change an implementation without affecting outside code.
Answer: A, D

Ques: 14 Given:
11. public static void test(String str) {
12. if (str == null | str.length() == 0) {
13. System.out.println("String is empty");
14. } else {
15. System.out.println("String is not empty");
16. }
17. }
And the invocation:
31. test(null);
What is the result?
A. An exception is thrown at runtime.
B. "String is empty" is printed to output.
C. Compilation fails because of an error in line 12.
D. "String is not empty" is printed to output.
Answer: A

Ques: 15 Given:
12. public class Wow {

13. public static void go(short n) {System.out.println("short");}
14. public static void go(Short n) {System.out.println("SHORT");}
15. public static void go(Long n) {System.out.println(" LONG");}
16. public static void main(String [] args) {
17. Short y = 6;
18. int z = 7;
19. go(y);
20. go(z);
21. }
22. }
What is the result?
A. short LONG
B. SHORT LONG
C. Compilation fails.
D. An exception is thrown at runtime.
Answer: C

Ques: 16 Click the Exhibit button.
Which three code fragments, added individually at line 29, produce the output 100? (Choose three.)

```
10. class Inner {
11.   private int x;
12.   public void setX( int x ) { this.x = x;
}
13.   public int getX() { return x; }
14. }
15.
16. class Outer {
17.   private Inner y;
18.   public void setY( Inner y ) { this.y =
y; }
19.   public Inner getY() { return y; }
20. }
21.
22. public class Gamma {
23.   public static void main( String[] args
) {
24.     Outer o = new Outer();
25.     Inner i = new Inner();
26.     int n = 10;
27.     i.setX( n );
28.     o.setY( i );
29.     // insert code here
30.   System.out.println( o.getY().getX() );
31.   }
32. }
```

A. n = 100;
B. i.setX(100);
C. o.getY().setX(100);
D. i = new Inner(); i.setX(100);
E. o.setY(i); i = new Inner(); i.setX(100);
F. i = new Inner(); i.setX(100); o.setY(i);

Answer: B, C, F

Ques: 17 Given:
10. class One {
11. public One() { System.out.print(1); }
12. }
13. class Two extends One {
14. public Two() { System.out.print(2); }
15. }
16. class Three extends Two {
17. public Three() { System.out.print(3); }
18. }
19. public class Numbers{
20. public static void main(String[] argv) { new Three(); }
21. }
What is the result when this code is executed?
A. 1
B. 3
C. 123
D. 321
E. The code runs with no output.
Answer: C

Ques: 18 Click the Exhibit button.
What is the outcome of the code?

```
 1. public class Item {
 2.   private String desc;
 3.   public String getDescription() { return
desc; }
 4.   public void setDescription(String d) {
desc = d; }
 5.
 6.   public static void modifyDesc(Item
item, String desc) {
 7.     item = new Item();
 8.     item.setDescription(desc);
 9.   }
10.   public static void main(String[] args)
{
11.     Item it = new Item();
12.     it.setDescription("Gobstopper");
13.     Item it2 = new Item();
14.     it2.setDescription("Fizzylifting");
15.     modifyDesc(it,
"Scrumdiddlyumptious");
16.
System.out.println(it.getDescription());
17.
System.out.println(it2.getDescription());
18.   }
19. }
```

A. Compilation fails.

B. Gobstopper
Fizzylifting
C. Gobstopper
Scrumdiddlyumptious
D. Scrumdiddlyumptious
Fizzylifting
E. Scrumdiddlyumptious
Answer: B

Ques: 19 Given:
10. class One {
11. public One foo() { return this; }
12. }
13. class Two extends One {
14. public One foo() { return this; }
15. }
16. class Three extends Two {
17. // insert method here
18. }
Which two methods, inserted individually, correctly complete the Three class? (Choose two.)
A. public void foo() {}
B. public int foo() { return 3; }
C. public Two foo() { return this; }
D. public One foo() { return this; }
E. public Object foo() { return this; }
Answer: C, D

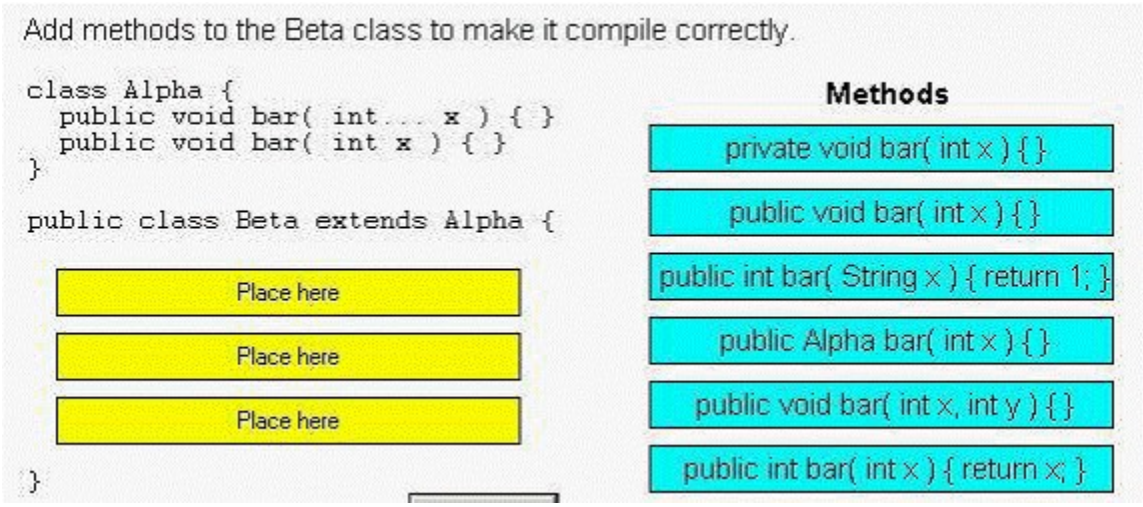
Ques: 20 Click the Exhibit button.
What is the output of the program shown in the exhibit?

```
10. class Foo {
11.  private int x;
12.  public Foo( int x ) { this.x = x; }
13.  public void setX( int x ) { this.x = x;
}
14.  public int getX() { return x; }
15. }
16.
17. public class Gamma {
18.
19.   static Foo fooBar( Foo foo ) {
20.     foo = new Foo( 100 );
21.     return foo;
22.   }
23.
24.   public static void main( String[] args
) {
25.   Foo foo = new Foo( 300 );
26.   System.out.print( foo.getX() + "-" );
27.
28.   Foo fooFoo = fooBar( foo );
29.   System.out.print( foo.getX() + "-" );
30.   System.out.print( fooFoo.getX() + "-"
);
31.
32.   foo = fooBar( fooFoo );
33.   System.out.print( foo.getX() + "-" );
34.   System.out.print( fooFoo.getX() );
35.   }
36. }
```

A. 300-100-100-100-100
B. 300-300-100-100-100
C. 300-300-300-100-100
D. 300-300-300-300-100

Answer: B

Ques: 21 Click the Task button.



Solution:

public void bar(int x){ }
public int bar(String x){ return 1; }

public void bar(int x,int y) { }

Ques: 22 Given:
10. interface A { public int getValue(); }
11. class B implements A {
12. public int getValue() { return 1; }
13. }
14. class C extends B {
15. // insert code here
16. }
Which three code fragments, inserted individually at line 15, make use of polymorphism? (Choose three.)
A. public void add(C c) { c.getValue(); }
B. public void add(B b) { b.getValue(); }
C. public void add(A a) { a.getValue(); }
D. public void add(A a, B b) { a.getValue(); }
E. public void add(C c1, C c2) { c1.getValue(); }
Answer: B, C, D

Ques: 23 Click the Exhibit button.
What is the result?

```
11. public class Bootchy {
12.   int bootch;
13.   String snootch;
14.
15.   public Bootchy() {
16.     this("snootchy");
17.     System.out.print("first ");
18.   }
19.
20.   public Bootchy(String snootch) {
21.     this(420, "snootchy");
22.     System.out.print("second ");
23.   }
24.
25.   public Bootchy(int bootch, String
snootch) {
26.     this.bootch = bootch;
27.     this.snootch = snootch;
28.     System.out.print("third ");
29.   }
30.
31.   public static void main(String[] args)
{
32.     Bootchy b = new Bootchy();
33.     System.out.print(b.snootch + " " +
b.bootch);
34.   }
35. }
```

A. snootchy 420 third second first
B. snootchy 420 first second third
C. first second third snootchy 420
D. third second first snootchy 420

E. third first second snootchy 420
F. first second first third snootchy 420
Answer: D

Ques: 24 Given:
10. abstract class A {
11. abstract void a1();
12. void a2() { }
13. }
14. class B extends A {
15. void a1() { }
16. void a2() { }
17. }
18. class C extends B { void c1() { } }
and:
A x = new B(); C y = new C(); A z = new C();
What are four valid examples of polymorphic method calls? (Choose four.)
A. x.a2();
B. z.a2();
C. z.c1();
D. z.a1();
E. y.c1();
F. x.a1();
Answer: A, B, D, F

Ques: 25 Click the Exhibit button.
What is the result?

```
1. public class SimpleCalc {
2.   public int value;
3.   public void calculate() { value += 7; }
4. }
```

And:

```
1. public class MultiCalc extends
SimpleCalc{
2.   public void calculate() { value -= 3; }
3.   public void calculate(int multiplier) {
4.     calculate();
5.     super.calculate();
6.     value *= multiplier;
7.   }
8.   public static void main(String[] args)
{
9.     MultiCalc calculator = new
MultiCalc();
10.    calculator.calculate(2);
11.    System.out.println("Value is: " +
calculator.value);
12.  }
13. }
```

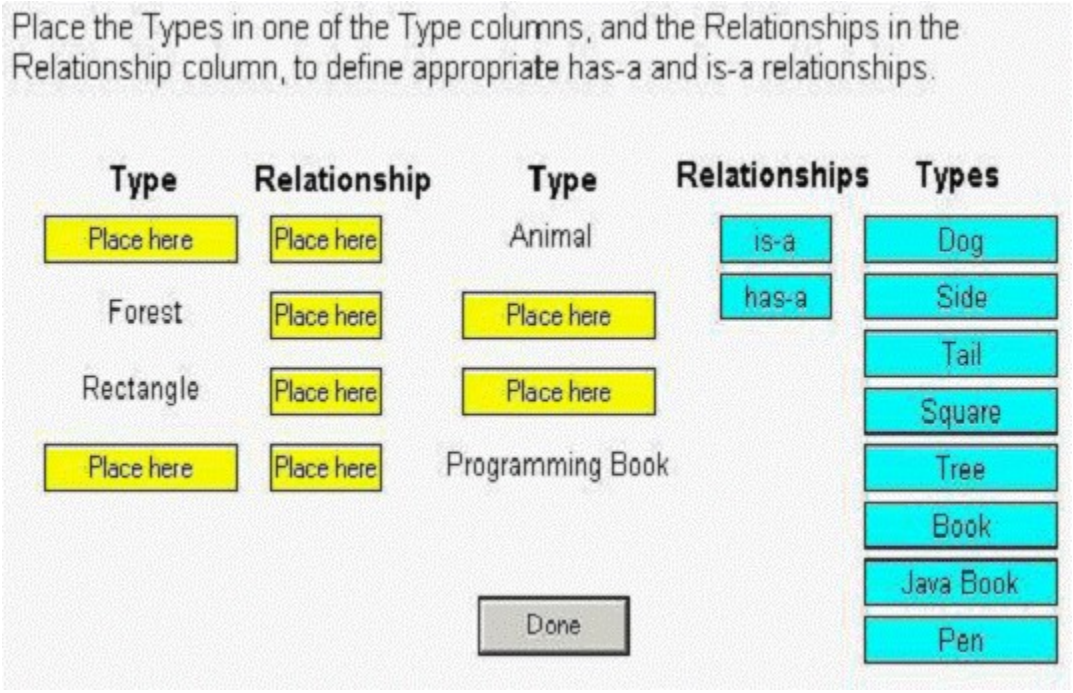
A. Value is: 8
B. Compilation fails.
C. Value is: 12
D. Value is: -12
E. The code runs with no output.
F. An exception is thrown at runtime.
Answer: A

Ques: 26 Given:
```
20. public class CreditCard {
21.
22. private String cardID;
23. private Integer limit;
24. public String ownerName;
25.
26. public void setCardInformation(String cardID,
27. String ownerName,
28. Integer limit) {
29. this.cardID = cardID;
30. this.ownerName = ownerName;
31. this.limit = limit;
32. }
33. }
```
Which statement is true?
A. The class is fully encapsulated.
B. The code demonstrates polymorphism.
C. The ownerName variable breaks encapsulation.
D. The cardID and limit variables break polymorphism.
E. The setCardInformation method breaks encapsulation.
Answer: C

Ques: 27Given:
```
11. class Animal { public String noise() { return "peep"; } }
12. class Dog extends Animal {
13. public String noise() { return "bark"; }
14. }
15. class Cat extends Animal {
16. public String noise() { return "meow"; }
17. }
...
30. Animal animal = new Dog();
31. Cat cat = (Cat)animal;
32. System.out.println(cat.noise());
```
What is the result?

A. peep
B. bark
C. meow
D. Compilation fails.
E. An exception is thrown at runtime.
Answer: E

Ques: 28 Click the Task button.



Solution:

Dog	**is-a**	**Animal**
Forest	**has-a**	**Tree**
Rectangle	**has-a**	**Side**
JavaBook	**is-a**	**ProgrammingBook**

Ques: 29 Which three statements are true? (Choose three.)
A. A final method in class X can be abstract if and only if X is abstract.
B. A protected method in class X can be overridden by any subclass of X.
C. A private static method can be called only within other static methods in class X.
D. A non-static public final method in class X can be overridden in any subclass of X.
E. A public static method in class X can be called by a subclass of X without explicitly referencing the class X.
F. A method with the same signature as a private final method in class X can be implemented in a subclass of X.
G. A protected method in class X can be overridden by a subclass of A only if the subclass is in the same
package as X.

Answer: B, E, F

Ques: 30 Which four statements are true? (Choose four.)
A. Has-a relationships should never be encapsulated.
B. Has-a relationships should be implemented using inheritance.
C. Has-a relationships can be implemented using instance variables.
D. Is-a relationships can be implemented using the extends keyword.
E. Is-a relationships can be implemented using the implements keyword.
F. The relationship between Movie and Actress is an example of an is-a relationship.
G. An array or a collection can be used to implement a one-to-many has-a relationship.
Answer: C, D, E, G

Ques: 31Given:
10. interface Jumper { public void jump(); }
...
20. class Animal {}
...
30. class Dog extends Animal {
31. Tail tail;
32. }
...
40. class Beagle extends Dog implements Jumper{
41. public void jump() {} 42. }
...
50. class Cat implements Jumper{
51. public void jump() {}
52. }
Which three are true? (Choose three.)
A. Cat is-a Animal
B. Cat is-a Jumper
C. Dog is-a Animal
D. Dog is-a Jumper
E. Cat has-a Animal
F. Beagle has-a Tail
G. Beagle has-a Jumper
Answer: B, C, F

Ques: 32 Click the Exhibit button.
What two must the programmer do to correct the compilation errors? (Choose two.)

```
1. public class Car {
2.   private int wheelCount;
3.   private String vin;
4.   public Car(String vin) {
5.     this.vin = vin;
6.     this.wheelCount = 4;
7.   }
8.   public String drive() {
9.     return "zoom-zoom";
10.  }
11.  public String getInfo() {
12.    return "VIN: " + vin + " wheels: " +
wheelCount;
13.  }
14. }
```

And:

```
1. public class MeGo extends Car {
2.   public MeGo(String vin) {
3.     this.wheelCount = 3;
4.   }
5. }
```

A. insert a call to this() in the Car constructor
B. insert a call to this() in the MeGo constructor
C. insert a call to super() in the MeGo constructor
D. insert a call to super(vin) in the MeGo constructor
E. change the wheelCount variable in Car to protected
F. change line 3 in the MeGo class to super.wheelCount = 3;
Answer: D, E

Ques: 33 Given:
10. public class SuperCalc {
11. protected static int multiply(int a, int b) { return a * b;}
12. }
and:
20. public class SubCalc extends SuperCalc{
21. public static int multiply(int a, int b) {
22. int c = super.multiply(a, b);
23. return c;
24. }
25. }
and:
30. SubCalc sc = new SubCalc ();
31. System.out.println(sc.multiply(3,4));
32. System.out.println(SubCalc.multiply(2,2));
What is the result?

A. 12
4
B. The code runs with no output.
C. An exception is thrown at runtime.
D. Compilation fails because of an error in line 21.
E. Compilation fails because of an error in line 22.
F. Compilation fails because of an error in line 31.
Answer: E

Ques: 34 Given:
1.public class Blip {
2. protected int blipvert(int x) { return 0; }
3. }
4. class Vert extends Blip {
5. // insert code here
6. }
Which five methods, inserted independently at line 5, will compile? (Choose five.)
A. public int blipvert(int x) { return 0; }
B. private int blipvert(int x) { return 0; }
C. private int blipvert(long x) { return 0; }
D. protected long blipvert(int x) { return 0; }
E. protected int blipvert(long x) { return 0; }
F. protected long blipvert(long x) { return 0; }
G. protected long blipvert(int x, int y) { return 0; }
Answer: A, C, E, F, G

Ques: 35 Given:
1. class Pizza {
2. java.util.ArrayList toppings;
3. public final void addTopping(String topping) {
4. toppings.add(topping);
5. }
6. }
7. public class PepperoniPizza extends Pizza {
8. public void addTopping(String topping) {
9. System.out.println("Cannot add Toppings");
10. }
11. public static void main(String[] args) {
12. Pizza pizza = new PepperoniPizza();
13. pizza.addTopping("Mushrooms");
14. }
15. }
What is the result?

Given:

```
10. public class Pizza {
11.   ArrayList toppings;
12.
13.   public final void addTopping(String
topping) {
14.     toppings.add(topping);
15.   }
16.
17.   public void removeTopping(String
topping) {
18.     toppings.remove(topping);
19.   }
20. }
```

And:

```
30. class PepperoniPizza extends Pizza {
31.  public void addTopping(String topping) {
32.    System.out.println("Cannot add
Toppings");
33.  }
34.
35.  public void removeTopping(String
topping) {
36.    System.out.println("Cannot remove
Pepperoni");
37.  }
38.}
```

And:

```
50. Pizza pizza = new PepperoniPizza();
51. pizza.addTopping("Mushrooms");
52. pizza.removeTopping("Pepperoni");
```

A. Compilation fails.
B. Cannot add Toppings
C. The code runs with no output.
D. A NullPointerException is thrown in Line 4.
Answer: A

Ques: 36 Given:

```
1. interface DoStuff2 {
2. float getRange(int low, int high); }
3.
4. interface DoMore {
5. float getAvg(int a, int b, int c); }
6.
7. abstract class DoAbstract implements DoStuff2, DoMore { }
8.
9. class DoStuff implements DoStuff2 {
10. public float getRange(int x, int y) { return 3.14f; } }
11.
12. interface DoAll extends DoMore {
```

13. float getAvg(int a, int b, int c, int d); }
What is the result?
A. The file will compile without error.
B. Compilation fails. Only line 7 contains an error.
C. Compilation fails. Only line 12 contains an error.
D. Compilation fails. Only line 13 contains an error.
E. Compilation fails. Only lines 7 and 12 contain errors.
F. Compilation fails. Only lines 7 and 13 contain errors.
G. Compilation fails. Lines 7, 12, and 13 contain errors.
Answer: A

Ques: 37 Given:
1. class TestA {
2. public void start() { System.out.println("TestA"); }
3. }
4. public class TestB extends TestA {
5. public void start() { System.out.println("TestB"); }
6. public static void main(String[] args) {
7. ((TestA)new TestB()).start();
8. }
9. }
What is the result?
A. TestA
B. TestB
C. Compilation fails.
D. An exception is thrown at runtime.
Answer: B

Ques: 38 Click the Exhibit button.
Given:
25. A a = new A();
26. System.out.println(a.doit(4, 5));
What is the result?

```
1. public class A {
2.   public String doit(int x, int y) {
3.     return "a";
4.   }
5.
6.   public String doit(int... vals) {
7.     return "b";
8.   }
9. }
```

A. Line 26 prints "a" to System.out.
B. Line 26 prints "b" to System.out.
C. An exception is thrown at line 26 at runtime.
D. Compilation of class A will fail due to an error in line 6.

Answer: A

Ques: 39 Given:

```
1. interface TestA { String toString(); }
2. public class Test {
3. public static void main(String[] args) {
4. System.out.println(new TestA() {
5. public String toString() { return "test"; }
6. });
7. }
8. }
```

What is the result?

A. test
B. null
C. An exception is thrown at runtime.
D. Compilation fails because of an error in line 1.
E. Compilation fails because of an error in line 4.
F. Compilation fails because of an error in line 5.

Answer: A

Ques: 40 Given:

```
11. public class ItemTest {
12. private final int id;
13. public ItemTest(int id) { this.id = id; }
14. public void updateId(int newId) { id = newId; }
15.
16. public static void main(String[] args) {
17. ItemTest fa = new ItemTest(42);
18. fa.updateId(69);
19. System.out.println(fa.id);
20. }
21. }
```

What is the result?

A. Compilation fails.
B. An exception is thrown at runtime.
C. The attribute id in the Item object remains unchanged.
D. The attribute id in the Item object is modified to the new value.
E. A new Item object is created with the preferred value in the id attribute.

Answer: A

Ques: 41 Given:

```
10. class One {
11. void foo() { }
```

12. }
13. class Two extends One {
14. //insert method here
15. }
Which three methods, inserted individually at line 14, will correctly complete class Two? (Choose three.)
A. int foo() { /* more code here */ }
B. void foo() { /* more code here */ }
C. public void foo() { /* more code here */ }
D. private void foo() { /* more code here */ }
E. protected void foo() { /* more code here */ }
Answer: B, C, E

Ques: 42 Given:
10. interface Data { public void load(); }
11. abstract class Info { public abstract void load(); }
Which class correctly uses the Data interface and Info class?
A. public class Employee extends Info implements Data {
public void load() { /*do something*/ }
}
B. public class Employee implements Info extends Data {
public void load() { /*do something*/ } }
C. public class Employee extends Info implements Data
public void load(){ /*do something*/ }
public void Info.load(){ /*do something*/ }
}
D. public class Employee implements Info extends Data {
public void Data.load(){ /*do something*/ }
public void load(){ /*do something*/ }
}
E. public class Employee implements Info extends Data {
public void load(){ /*do something*/ }
public void Info.load(){ /*do something*/ }
}
F. public class Employee extends Info implements Data{
public void Data.load() { /*do something*/ }
public void Info.load() { /*do something*/ } }
Answer: A

Ques: 43 Given:
11. public abstract class Shape {
12. int x;
13. int y;
14. public abstract void draw();

15. public void setAnchor(int x, int y) {
16. this.x = x;
17. this.y = y;
18. }
19. }
and a class Circle that extends and fully implements the Shape class.
Which is correct?
A. Shape s = new Shape();
s.setAnchor(10,10);
s.draw();
B. Circle c = new Shape();
c.setAnchor(10,10);
c.draw();
C. Shape s = new Circle();
s.setAnchor(10,10);
s.draw();
D. Shape s = new Circle();
s->setAnchor(10,10);
s->draw();
E. Circle c = new Circle();
c.Shape.setAnchor(10,10);
c.Shape.draw();
Answer: C

Ques: 44 Click the Exhibit button.
What is the result?

```
11. class Person {
12.   String name = "No name";
13.   public Person(String nm) { name = nm; }
14. }
15.
16. class Employee extends Person {
17.   String empID = "0000";
18.   public Employee(String id) { empID =
id; }
19. }
20.
21. public class EmployeeTest {
22.   public static void main(String[] args)
{
23.     Employee e = new Employee("4321");
24.     System.out.println(e.empID);
25.   }
26. }
```

A. 4321
B. 0000
C. An exception is thrown at runtime.
D. Compilation fails because of an error in line 18.
Answer: D

Ques: 45 Given:
10. interface Foo {}
11. class Alpha implements Foo {}
12. class Beta extends Alpha {}
13. class Delta extends Beta {
14. public static void main(String[] args) {
15. Beta x = new Beta();
16. // insert code here
17. }
18. }
Which code, inserted at line 16, will cause a java.lang.ClassCastException?
A. Alpha a = x;
B. Foo f = (Delta)x;
C. Foo f = (Alpha)x;
D. Beta b = (Beta)(Alpha)x;
Answer: B

Ques: 46 Given:
1. class SuperClass {
2. public A getA() {
3. return new A();
4. }
5. }
6. class SubClass extends SuperClass {
7. public B getA(){
8. return new B();
9. }
10. }
Which statement is true?
A. Compilation will succeed if A extends B.
B. Compilation will succeed if B extends A.
C. Compilation will always fail because of an error in line 7.
D. Compilation will always fail because of an error in line 8.
Answer: B

Ques: 47 Given:
11. static class A {
12. void process() throws Exception { throw new Exception(); }
13. }
14. static class B extends A {
15. void process() { System.out.println("B"); }
16. }
17. public static void main(String[] args) {

18. new B().process();
19. }
What is the result?
A. B
B. The code runs with no output.
C. Compilation fails because of an error in line 12.
D. Compilation fails because of an error in line 15.
E. Compilation fails because of an error in line 18.
Answer: A

Ques: 48 Given:
11. class ClassA {}
12. class ClassB extends ClassA {}
13. class ClassC extends ClassA {}
and:
21. ClassA p0 = new ClassA();
22. ClassB p1 = new ClassB();
23. ClassC p2 = new ClassC();
24. ClassA p3 = new ClassB();
25. ClassA p4 = new ClassC();
Which three are valid? (Choose three.)
A. p0 = p1;
B. p1 = p2;
C. p2 = p4;
D. p2 = (ClassC)p1;
E. p1 = (ClassB)p3;
F. p2 = (ClassC)p4;
Answer: A, E, F

Ques: 49 Given:
11. abstract class Vehicle { public int speed() { return 0; }
12. class Car extends Vehicle { public int speed() { return 60; }
13. class RaceCar extends Car { public int speed() { return 150; }
...
21. RaceCar racer = new RaceCar();
22. Car car = new RaceCar();
23. Vehicle vehicle = new RaceCar();
24. System.out.println(racer.speed() + ", " + car.speed()
25. + ", " + vehicle.speed());
What is the result?
A. 0, 0, 0
B. 150, 60, 0
C. Compilation fails.
D. 150, 150, 150

E. An exception is thrown at runtime.
Answer: D

Ques: 50 Given code in separate source files:
10. public class Foo {
11. public int a;
12. public Foo() { a = 3; }
13. public void addFive() { a += 5;}
14. } and: 20. public class Bar extends Foo {
21. public int a;
22. public Bar() { a = 8; }
23. public void addFive() { this.a += 5; }
24. } invoked with:
30. Foo foo = new Bar();
31. foo.addFive();
32. System.out.println("Value: " + foo.a);
What is the result?
A. Value: 3
B. Value: 8
C. Value: 13
D. Compilation fails.
E. The code runs with no output.
F. An exception is thrown at runtime.
Answer: A

Ques: 51 Which Man class properly represents the relationship "Man has a best friend who is a Dog"?
A. class Man extends Dog { }
B. class Man implements Dog { }
C. class Man { private BestFriend dog; }
D. class Man { private Dog bestFriend; }
E. class Man { private Dog<bestFriend>; }
F. class Man { private BestFriend<dog>; }
Answer: D

Ques: 52 Given:
10. interface Foo { int bar(); }
11. public class Sprite {
12. public int fubar(Foo foo) { return foo.bar(); }
13. public void testFoo() {
14. fubar(
15. // insert code here
16.);
17. }

18. }

Which code, inserted at line 15, allows the class Sprite to compile?

A. Foo { public int bar() { return 1; }

B. new Foo { public int bar() { return 1; }

C. new Foo() { public int bar() { return 1; }

D. new class Foo { public int bar() { return 1; }

Answer: C

Ques: 53 Click the Task button.

Given:

```
public class Doubler {
  public static int doubleMe( Holder h) {
    return h.getAmount() * 2;
  }
}
```

and:

```
public class Holder {
  int amount = 10;
  public void doubleAmount(){ amount = Doubler.doubleMe( this );}
  public int getAmount(){ return amount;}
  //more code here
}
```

Place the code fragments in position to reduce the coupling between Doubler and Holder.

```
public class Doubler {
  public static int doubleMe( [Place here]  h) {
    return [Place here] * 2;
  }
}

public class Holder {
  int amount = 10;
  public void doubleAmount(){ amount = Doubler.doubleMe( [Place here] );}
  public int getAmount(){ return amount;}
  //more code here
}
```

Code Fragments

void	Holder	int	Doubler
h.getAmount()	h	this	amount

Done

Solution:

1. int 2. h 3. amount.

Ques: 54 Given classes defined in two different files:

1. package packageA;

2. public class Message {

3. String getText() { return "text"; }

4. }

and:

1. package packageB;

2. public class XMLMessage extends packageA.Message {

3. String getText() { return "<msg>text</msg>";}
4. public static void main(String[] args) {
5. System.out.println(new XMLMessage().getText());
6. }
7. }
What is the result of executing XMLMessage.main?
A. text
B. An exception is thrown at runtime.
C. Compilation fails because of an error in line 2 of XMLMessage.
D. Compilation fails because of an error in line 3 of XMLMessage.
Answer: D

Ques: 55 Given:
1. public class A {
2. public void doit() {
3. }
4. public String doit() {
5. return "a";
6. }
7. public double doit(int x) {
8. return 1.0;
9. }
10. }
What is the result?
A. An exception is thrown at runtime.
B. Compilation fails because of an error in line 7.
C. Compilation fails because of an error in line 4.
D. Compilation succeeds and no runtime errors with class A occur.
Answer: C

Ques: 56
Click the Exhibit button.
1. public class GoTest {
2. public static void main(String[] args) {
3. Sente a = new Sente(); a.go();
4. Goban b = new Goban(); b.go();
5. **Stone c = new Stone(); c.go();**
6. **}**
7. **}**
8.
9. **class Sente implements Go {**
10. public void go() { System.out.println(”go in Sente.”); }
11. }
12.

13. class Goban extends Sente {
14. public void go() { System.out.println(”go in Goban”); }
15. **}**
16.
17. class Stone extends Goban implements Go { }
18.
19. interface Go { public void go(); }
What is the result?
A. go in Goban
go in Sente
go in Sente
B. go in Sente
go in Sente
go in Goban
C. go in Sente
go in Goban
go in Goban
D. go in Goban
go in Goban
go in Sente
E. Compilation fails because of an error in line 17.
Answer: C

Ques: 57
Click the Exhibit button.
1. public class Employee {
2. String name;
3. double baseSalary;
4. Employee(String name, double baseSalary) {
5. this.name = name;
6. this.baseSalary = baseSalary;
7. }
8. }
And:
1. public class Salesperson extends Employee {
2. double commission;
3. public Salesperson(String name, double baseSalary,
4. double commission) {
5. // insert code here
6. }
7. }
Which code, inserted at line 7, completes the Salesperson constructor?
A. this.commission = commission;
B. superb();

commission = commission;
C. this.commission = commission;
superb();
D. super(name, baseSalary);
this.commission = commission;
E. super();
this.commission = commission;
F. this.commission = commission;
super(name, baseSalary);
Answer: D

Ques: 58
Given:
1. public class Team extends java.util.LinkedList {
2. public void addPlayer(Player p) {
3. add(p);
4. }

5. **public void compete(Team opponent) { /* more code here */ }**
6. }
7. class Player { /* more code here */ }
Which two are true? (Choose two.)
A. This code will compile.
B. This code demonstrates proper design of an is-a relationship.
C. This code demonstrates proper design of a has-a relationship.
D. A Java programmer using the Team class could remove Player objects from a Team object.
Answer: AD

Ques: 59
Click the Exhibit button.
11. class Payload {
12. private int weight;
13. public Payload(int wt) { weight = wt; }
13. public void setWeight(mt w) { weight = w; }
15. public String toString { return Integer.toString(weight); }
16. }
17.
18. public class TestPayload {
19. static void changePayload(Payload p) {
20. /* insert code here */
21. }
22.
23. public static void main(String[] args) {

24. Payload p = new Payload();
25. p.setWeight(1024);
26. changePayload(p);
27. System.out.println(”The value of p is “+ p);
28. }
29. }
Which statement, placed at line 20, causes the code to print “The value of p is 420.”?
A. p.setWeight(420);
B. p.changePayload(420);
C. p = new Payload(420);
D. Payload.setWeight(420);
E. p = Payload.setWeight(420);
F. p = new Payload();
p.setWeight(420);
Answer: A

Ques: 60. Which statement(s) are true? (Choose all that apply.)
A. Has-a relationships always rely on inheritance.
B. Has-a relationships always rely on instance variables. C. Has-a relationships always require at least two class types.
D. Has-a relationships always rely on polymorphism.
E. Has-a relationships are always tightly coupled.
Answer:
-> **B** is correct.
-> **A** and **D** describe other OO topics. **C** is incorrect because a class can have an instance of itself. **E** is incorrect because while has-a relationships can lead to tight coupling, it is by no means *always* the case.

Ques: 61. Given:
```
class Clidders {
public final void flipper() { System.out.println("Clidder"); }
}
public class Clidlets extends Clidders {
public void flipper() {
System.out.println("Flip a Clidlet");
super.flipper();
}
public static void main(String [] args) {
new Clidlets().flipper();
}
}
```
What is the result?
A. Flip a Clidlet

B. Flip a Clidder
C. Flip a Clidder
Flip a Clidlet
D. Flip a Clidlet
Flip a Clidder
E. Compilation fails.

Answer:

-> **E** is correct. final methods cannot be overridden.
->**A, B, C,** and **D** are incorrect based on the above.

Ques: 62. Given:
public abstract interface Frobnicate { public void twiddle(String s); }
Which is a correct class? (Choose all that apply.)
A. public abstract class Frob implements Frobnicate {
public abstract void twiddle(String s) { }
}
B. public abstract class Frob implements Frobnicate { }
C. public class Frob extends Frobnicate {
public void twiddle(Integer i) { }
}
D. public class Frob implements Frobnicate {
public void twiddle(Integer i) { }
}
E. public class Frob implements Frobnicate {
public void twiddle(String i) { }
public void twiddle(Integer s) { }
}

Answer:

->**B** is correct, an abstract class need not implement any or all of an interface's methods. **E** is correct, the class implements the interface method and additionally overloads the twiddle() method.
-> **A** is incorrect because abstract methods have no body.
C is incorrect because classes implement interfaces they don't extend them.
D is incorrect because overloading a method is not implementing it.

Ques: 63. Given:

```
class Top {
public Top(String s) { System.out.print("B"); }
}
public class Bottom2 extends Top {
public Bottom2(String s) { System.out.print("D"); }
public static void main(String [] args) {
new Bottom2("C");
```

System.out.println(" ");
} }
What is the result?
A. BD
B. DB
C. BDC
D. DBC
E. Compilation fails.
Answer:
-> **E** is correct. The implied super() call in Bottom2's constructor cannot be satisfied because there isn't a no-arg constructor in Top. A default, no-arg constructor is generated by the compiler only if the class has no constructor defined explicitly.
-> **A, B, C,** and **D** are incorrect based on the above.

Ques: 64. Select the two statements that best indicate a situation with low coupling. (Choose two.)
A. The attributes of the class are all private.
B. The class refers to a small number of other objects.
C. The object contains only a small number of variables.
D. The object is referred to using an anonymous variable, not directly.
E. The reference variable is declared for an interface type, not a class. The interface provides a small number of methods.
F. It is unlikely that changes made to one class will require any changes in another.
Answer:
-> **E** and **F** are correct. Only having access to a small number of methods implies limited coupling. If the access is via a reference of interface type, it may be argued that there is even less opportunity for coupling as the class type itself is not visible. Stating that changes in one part of a program are unlikely to cause consequences in another part is really the essence of low coupling. There is no such thing as an anonymous variable. Referring to only a small number of other objects might imply low coupling, but if each object has many methods, and all are used, then coupling is high. Variables (attributes) in a class should usually be private, but this describes encapsulation, rather than low coupling. Of course, good encapsulation tends to reduce coupling as a consequence.
-> **A, B, C** and **D** are incorrect based on the preceding treatise.

Ques: 65. Given:
class Clidder {
private final void flipper() { System.out.println("Clidder"); }

}
public class Clidlet extends Clidder {
public final void flipper() { System.out.println("Clidlet"); }
public static void main(String [] args) {
new Clidlet().flipper();

} }
What is the result?
A. Clidlet
B. Clidder
C. Clidder
Clidlet
D. Clidlet
Clidder
E. Compilation fails.
Answer:
-> **A** is correct. Although a final method cannot be overridden, in this case, the method is private, and therefore hidden. The effect is that a new, accessible, method flipper is created. Therefore, no polymorphism occurs in this example, the method invoked is simply that of the child class, and no error occurs.
-> **B, C, D,** and **E** are incorrect based on the preceding.

Ques: 66. Given:
1. class Plant {
2. String getName() { return "plant"; }
3. Plant getType() { return this; }
4. }
5. class Flower extends Plant {
6. // insert code here
7. }
8. class Tulip extends Flower { }
Which statement(s), inserted at line 6, will compile? (Choose all that apply.)
A. Flower getType() { return this; }
B. String getType() { return "this"; }
C. Plant getType() { return this; }
D. Tulip getType() { return new Tulip(); }
Answer:
-> **A, C,** and **D** are correct. **A** and **D** are examples of co-variant returns, i.e., Flower and Tulip are both subtypes of Plant.
->**B** is incorrect, String is not a subtype of Plant.

Ques: 67. Given:
1. class Zing {
2. protected Hmpf h;
3. }
4. class Woop extends Zing { }
5. class Hmpf { }
Which is true? (Choose all that apply.)
A. Woop is-a Hmpf and has-a Zing.
B. Zing is-a Woop and has-a Hmpf.
C. Hmpf has-a Woop and Woop is-a Zing.
D. Woop has-a Hmpf and Woop is-a Zing.

E. Zing has-a Hmpf and Zing is-a Woop.

Answer:

-> **D** is correct, Woop inherits a Hmpf from Zing.

-> **A, B, C,** and **E** are incorrect based on the preceding.

Ques: 68. Given:

```
1. class Programmer {
2. Programmer debug() { return this; }
3. }
4. class SCJP extends Programmer {
5. // insert code here
6. }
```

Which, inserted at line 5, will compile? (Choose all that apply.)

A. Programmer debug() { return this; }
B. SCJP debug() { return this; }
C. Object debug() { return this; }
D. int debug() { return 1; }
E. int debug(int x) { return 1; }
F. Object debug(int x) { return this; }

Answer:

-> **A, B, E,** and **F** are correct. **A** and **B** are examples of overriding, specifically, **B** is an example of overriding using a covariant return. **E** and **F** are examples of overloading.

-> **C** and **D** are incorrect. They are illegal overrides because their return types are incompatible. They are illegal overloads because their arguments did not change.

Ques: 69. Given:

```
class Uber {
static int y = 2;
Uber(int x) { this(); y = y * 2; }
Uber() { y++; }
}
class Minor extends Uber {
Minor() { super(y); y = y + 3; }
public static void main(String [] args) {
new Minor();
System.out.println(y);
} }
```

What is the result?

A. 6 B. 7
C. 8 D. 9
E. Compilation fails. F. An exception is thrown.

Answer:

-> **D** is correct. Minor's constructor makes an explicit call to Uber's 1-arg constructor, which makes an explicit (this) call to Uber's no-arg constructor, which increments y,

then returns to the 1-arg constructor, which multiples y * 2, and then returns to Minor's constructor, which adds 3 to y.
-> **A, B, C, E,** and **F** are incorrect based on the preceding.

Ques: 70. Which statement(s) are true? (Choose all that apply.)
A. Cohesion is the OO principle most closely associated with hiding implementation details.
B. Cohesion is the OO principle most closely associated with making sure that classes know about other classes only through their APIs.
C. Cohesion is the OO principle most closely associated with making sure that a class is designed with a single, well-focused purpose.
D. Cohesion is the OO principle most closely associated with allowing a single object to be seen as having many types.
Answer:
-> Answer **C** is correct.
-> **A** refers to encapsulation, **B** refers to coupling, and **D** refers to polymorphism.

Ques: 71. Given:
```
1. class Dog { }
2. class Beagle extends Dog { }
3.
4. class Kennel {
5. public static void main(String [] arfs) {
6. Beagle b1 = new Beagle();
7. Dog dog1 = new Dog();
8. Dog dog2 = b1;
9. // insert code here
10. }
11. }
```
Which, inserted at line 9, will compile? (Choose all that apply.)
A. Beagle b2 = (Beagle) dog1; B. Beagle b3 = (Beagle) dog2;
C. Beagle b4 = dog2; D. None of the above statements will compile
Answer:
-> **A** and **B** are correct. However, at runtime, **A** will throw a ClassCastException because dog1 refers to a Dog object, which can't necessarily do Beagle stuff.
-> **C** and **D** are incorrect based on the preceding.

Ques: 72. Given the following,
```
1. class X { void do1() { } }
2. class Y extends X { void do2() { } }
3.
4. class Chrome {
5. public static void main(String [] args) {
6. X x1 = new X();
7. X x2 = new Y();
```

8. Y y1 = new Y();
9. // insert code here
10. }
11. }
Which, inserted at line 9, will compile? (Choose all that apply.)
A. x2.do2(); B. (Y)x2.do2();
C. ((Y)x2).do2(); D. None of the above statements will compile.
Answer:
-> **C** is correct. Before you can invoke Y's do2 method you have to cast x2 to be of type Y. Statement **B** looks like a proper cast but without the second set of parentheses, the compiler thinks it's an incomplete statement.
-> **A, B** and **D** are incorrect based on the preceding.

Ques: 73. Given:
```
class Bird {
{ System.out.print("b1 "); }
public Bird() { System.out.print("b2 "); }
}
class Raptor extends Bird {
static { System.out.print("r1 "); }
public Raptor() { System.out.print("r2 "); }
{ System.out.print("r3 "); }
static { System.out.print("r4 "); }
}
class Hawk extends Raptor {
public static void main(String[] args) {
System.out.print("pre ");
new Hawk();
System.out.println("hawk ");
}
}
```
What is the result?
A. pre b1 b2 r3 r2 hawk B. pre b2 b1 r2 r3 hawk
C. pre b2 b1 r2 r3 hawk r1 r4 D. r1 r4 pre b1 b2 r3 r2 hawk
E. r1 r4 pre b2 b1 r2 r3 hawk F. pre r1 r4 b1 b2 r3 r2 hawk
G. pre r1 r4 b2 b1 r2 r3 hawk H. The order of output cannot be predicted.
I. Compilation fails.
Answer:
-> **D** is correct. Static init blocks are executed at class loading time, instance init blocks run right after the call to super() in a constructor. When multiple init blocks of a single type occur in a class, they run in order, from the top down.
-> **A, B, C, E, F, G, H,** and **I** are incorrect based on the above.