[Declarations, Initialization and Scoping](javascript:setOfferingID(12);setActionType('load');setOfferingType(0);submit())

Question: 1 / 9

Which of the following are true ?

Choose: 1

Options

1.) final abstract methods only belong in an interface and nowhere else

2.) static abstract methods only belong in an interface

3.) static public methods of a class B can be overriden by a subclass of B

4.) An interface's members variables are public static final by default

5.) None of the above

Question: 2 / 9   
  
Choose: 1   
  
Options  
  
1.) 10   
  
2.) 20   
  
3.) 11   
  
4.) 21   
  
5.) None of the above

Question: 3 / 9   
  
Choose: 1   
  
Options  
  
1.) 25   
  
2.) 5   
  
3.) 20   
  
4.) Runtime error   
  
5.) Compile error

Question: 4 / 9   
  
Choose: 1   
  
Options  
  
1.) public boolean isTankFull(){return tankFull;} public void setTankFull(boolean tabkFull){this.tankFull = tabkFull;}   
  
2.) public boolean getTankFull(){return tankFull;} public void setTankFull(boolean tabkFull){this.tankFull = tabkFull;}   
  
3.) public boolean getTankFull(boolean tankFull){return tankFull;} public void setTankFull(boolean tabkFull){this.tankFull = tabkFull;}   
  
4.) public boolean gettankFull(){return tankFull;} public void settankFull(boolean tabkFull){this.tankFull = tabkFull;}   
  
5.) None of the above

Question: 5 / 9   
  
Choose: 4   
  
Options  
  
1.) Shackle.main() is a valid invocation   
  
2.) Shackle.go() is a valid invocation   
  
3.) new Shackle().main() is a valid invocation   
  
4.) new Shackle().go() is a valid invocation   
  
5.) go() can be called from main() directly   
  
6.) main() can be called from go() directly

Question: 6 / 9   
  
Choose: 3   
  
Options  
  
1.) public class Shackle extends Leg { public void grow() {} }   
  
2.) public abstract class Shackle extends Leg { public void grow() {} }   
  
3.) public abstract class Shackle extends Leg { public abstract void grow(); }   
  
4.) public class Shackle extends Leg { public void grow(); }   
  
5.) public class Shackle extends Leg { public void grow(int i){}; }

Question: 7 / 9   
  
Choose: 1   
  
Options  
  
1.) NullpointerException 0   
  
2.) NullpointerException NullpointerException   
  
3.) 0 0   
  
4.) Compile error   
  
5.) 0 NullPointerException   
  
6.) None of the above

Question: 8 / 9   
  
Choose: 2   
  
Options  
  
1.) #1 will always return true   
  
2.) #2 will always return true   
  
3.) #3 will not compile   
  
4.) #4 will return true   
  
5.) #5 will return true   
  
6.) #4 will not compile   
  
7.) #3 will return true

Question: 9 / 9   
  
Choose: 2   
  
Options  
  
1.) Glad IS A mad   
  
2.) Glad IS A Serializable   
  
3.) An interface cannot extend more than one interface. Glad will not compile   
  
4.) None of the above

Flow Control

Question: 1 / 9   
  
Choose: 1   
  
Options  
  
1.) x++;   
  
2.) x = x;   
  
3.) x--;   
  
4.) x = x+2;   
  
5.) x = x + 3;   
  
6.) No code fragment will help as this program will not compile   
  
7.) This program will compile but will result in a runtime error

Question: 5 / 9   
  
Choose: 1   
  
Options  
  
1.) one five   
  
2.) two   
  
3.) three five   
  
4.) five   
  
5.) four   
  
6.) None of the above

Question: 6 / 9   
  
Choose: 3   
  
Options  
  
1.) A switch statement can contain enums   
  
2.) An if statement can have multiple 'else if' statements following it   
  
3.) The default keyword must be the last one used in a switch statement   
  
4.) An Error and an Exception can both be captured by catching a Throwable   
  
5.) Checked exceptions are those that extend RuntimeException

Question: 9 / 9   
  
Choose: 1   
  
Options  
  
1.) 10   
  
2.) 11   
  
3.) 0   
  
4.) Compile time error   
  
5.) Runtime error

Question: 5 / 8   
  
Choose: 1   
  
Options  
  
1.) 1 2 3 4   
  
2.) 1 2 3   
  
3.) 2 3 4   
  
4.) 3 4 5   
  
5.) 2 3 4 5   
  
6.) None of the above

Question: 6 / 8   
  
Choose: 1   
  
Options  
  
1.) 10   
  
2.) 11   
  
3.) 12   
  
4.) None of the above

Question: 7 / 8   
  
Choose: 2   
  
Options  
  
1.) This is an enhanced for loop   
  
2.) i may be an Integer[], Object[], iterator, or a Comparable   
  
3.) There is more than one syntax for this loop   
  
4.) for (Object o : list) where list is an ArrayList will compile   
  
5.) for (Object o : list.iterator()) where list is an ArrayList will compile   
  
6.) This loop is available in java 1.4 and above

Questions no -1

What is the output for the below code ?

public class Test{

public static void main(String[] args) {

int i1=1;

switch(i1){

case 1:

System.out.println("one");

case 2:

System.out.println("two");

case 3:

System.out.println("three");

}

}

}

options

A)one two three

B)one

C)one two

D)Compile error.

Questions no -2

What is the output for the below code ?

public class Test {

public static void main(String[] args) {

char c = 'a';

switch(c){

case 65:

System.out.println("one");break;

case 'a':

System.out.println("two");break;

case 3:

System.out.println("three");

}

}

}

options

A)one two three

B)one

C)two

D)Compile error - char can't be in switch statement.

01

Given the code. What is the result?

**class** Small {  
    **public** Small() {  
        System.out.print("a ");  
        **super**();  
    }  
}  
  
**class** Small2 **extends** Small {  
    **public** Small2() {  
        System.out.print("b ");  
        **super**();  
    }  
}  
  
**class** Small3 **extends** Small2 {  
    **public** Small3() {  
        System.out.print("c ");  
        **super**();  
    }  
}  
  
**public** **class** Test {       
    **public** **static** void main(String args[]) {  
        **new** Small3();  
    }  
}

|  |
| --- |
| A) a |
| B) c |
| C) a b c |
| D) c b a |
| E) compilation fails. |
| F) The code runs without output. |

02

Given the code. What is the result?

**public** **class** EmptyStringsTest {  
    **public** **static** boolean isEmpty(String s) {  
        **return** (s == null | s.length() == 0);  
    }  
  
    **public** **static** void main(String args[]) {  
        **if** (isEmpty(null)) {  
            System.out.print("empty ");  
        } **else** {  
            System.out.print("not\_empty ");  
        }  
    }  
}

|  |
| --- |
| A) "empty" is printed |
| B) "not\_empty" is printed |
| C) Compilation fails |
| D) An exception is thrown at runtime |

03

Given the code. What is the result after the class TryMe execution?

**class** A {  
    **public** void doA() {  
        B b = **new** B();  
        b.dobB();  
        System.out.print("doA");  
    }  
}  
  
**class** B {  
    **public** void dobB() {  
        C c = **new** C();  
        c.doC();  
        System.out.print("doB");  
    }  
}  
  
**class** C {  
    **public** void doC() {  
        **if** (true)  
            **throw** **new** NullPointerException();  
        System.out.print("doC");  
    }  
}  
  
**public** **class** TryMe {  
  
    **public** **static** void main(String args[]) {  
        **try** {  
            A a = **new** A();  
            a.doA();  
        } **catch** (Exception ex) {  
            System.out.print("error");  
        }  
    }  
}

|  |
| --- |
| A) "doCdoBdoA" is printed |
| B) "doAdoBdoC" is printed |
| C) "doBdoAerror" is printed |
| D) "error" is printed |
| E) nothing is printed |

04

Give the code. What is the result?

**class** Hotel {  
    **public** int bookings;  
    **public** void book() {  
        bookings++;  
    }  
}  
  
**public** **class** SuperHotel **extends** Hotel {  
    **public** void book() {  
        bookings--;  
    }  
      
    **public** void book(int size) {  
        book();  
        **super**.book();  
        bookings += size;  
    }  
      
    **public** **static** void main(String args[]) {  
        SuperHotel hotel = **new** SuperHotel();  
        hotel.book(2);  
        System.out.print(hotel.bookings);  
    }  
}

|  |
| --- |
| A) Compilation fails. |
| B) An exception is thrown at runtime. |
| C) 0 |
| D) 1 |
| E) 2 |
| F) -1 |

05

Given the code. What is the result?

1.      int i = 10;  
2.      **while** (i++ <= 10) {  
3.          i++;  
4.      }  
5.      System.out.print(i);

|  |
| --- |
| A) 10 |
| B) 11 |
| C) 12 |
| D) 13 |
| E) Line 5 will be never reached. |

06

Given the code. What is the result?

1.  **public** **static** void main(String args[]) {  
2.      Object myObj = **new** String[]{"one", "two", "three"} {  
3.          **for** (String s : (String[])myObj) System.out.print(s + ".");  
4.      }  
5.  }

|  |
| --- |
| A) one.two.three. |
| B) Compilation fails because of an error at line 2 |
| C) Compilation fails because of an error at line 3 |
| D) An exception is thrown at runtime. |

07

Given the code. What is the result?

**class** Vehicle {  
    **public** void printSound() {  
        System.out.print("vehicle");  
    }  
}  
  
**class** Car **extends** Vehicle {  
    **public** void printSound() {  
        System.out.print("car");  
    }  
}  
  
**class** Bike **extends** Vehicle {  
    **public** void printSound() {  
        System.out.print("bike");  
    }  
}  
  
**public** **class** Test {  
    **public** **static** void main(String[] args) {  
        Vehicle v = **new** Car();  
        Bike b = (Bike) v;  
          
        v.printSound();  
        b.printSound();  
    }     
}

|  |
| --- |
| A) Compilation fails. |
| B) An exception is thrown at runtime. |
| C) "vehiclecar" is printed. |
| D) "vehiclebike" is printed. |
| E) "carcar" is printed. |
| F) "bikebike" is printed |

08

Given the code. What is the output?

**public** **class** Test {       
    int a = 10;  
      
    **public** void doStuff(int a) {  
        a += 1;  
        System.out.println(++a);  
    }  
    **public** **static** void main(String args[]) {  
        Test t = **new** Test();  
        t.doStuff(3);  
    }  
}

|  |
| --- |
| A) 4 |
| B) 5 |
| C) 12 |
| D) 11 |

10

Which code, inserted inserted at line labeled "//some code goes her", allows the class Test to be compiled?

**class** Util {  
    **public** enum State{ACTIVE, DELETED, INACTIVE}  
}  
  
**public** **class** Test {       
    **public** **static** void main(String args[]) {  
        //some code goes here         
    }  
}

|  |
| --- |
| A) State state = State.INACTIVE; |
| B) State state = INACTIVE; |
| C) Util.State state = Util.State.INACTIVE; |
| D) State state = Util.INACTIVE; |

11

Given the code. Which statements are true? (Select two)

**public** **class** Hotel {           
  
        **public** **static** void book() {  
                //some code goes here  
        }  
          
        **public** void cancelBooking() {  
                //some code goes here  
        }         
}

|  |
| --- |
| A) Method book() can directly call method cancelBooking() |
| B) Method cancelBooking() can directly call method book() |
| C) Hotel.book() is a valid invocation of book() |
| D) Hotel.cancelBooking() is a valid invocation of cancelBooking() |

12

Given the exhibit. What is the result?

**public** **class** Hotel {  
      
    **public** **static** void book(short a) {  
        System.out.print("short ");  
    }  
      
    **public** **static** void book(Short a) {  
        System.out.print("SHORT ");  
    }  
      
    **public** **static** void book(Long a) {  
        System.out.print("LONG ");  
    }  
      
    **public** **static** void main(String[] args) {  
        short shortRoom = 1;  
        int intRoom = 2;  
          
        book(shortRoom);  
        book(intRoom);  
    }  
}

|  |
| --- |
| A) SHORT LONG |
| B) short LONG |
| C) Compilation fails |
| D) An exception is thrown at runtime |

13

What do you need to do to correct compilation errors? (Select two)

**public** **class** Creature {  
    **private** int legCount;  
    **private** int wingCount;  
      
    **public** Creature(int legCount) {  
        **this**.legCount = **this**.legCount;  
        **this**.wingCount = 0;  
    }  
      
    **public** String toString() {  
        **return** "legs=" + **this**.legCount + " wings=" + wingCount;  
    }  
}  
  
**public** **class** Animal **extends** Creature {  
    **public** Animal(int legCount) {  
        **this**.wingCount = 0;  
    }  
  
}

|  |
| --- |
| A) insert a call to super() into Creature constructor. |
| B) insert a call to super() into Animal constructor. |
| C) insert a call to this() into Animal constructor. |
| D) insert a call to super(legCount) into Animal constructor. |
| E) change the wingCount variable in the class Creature to protected. |
| F) change the string "this.wingCount = 0" in the class Animal to "super.wingCount = 0" |

14

What can directly access and change the value of the variable roomNr?

**package** com.mycompany;  
  
**public** **class** Hotel  {  
    **private** int roomNr = 100;  
}

|  |
| --- |
| A) Only the Hotel class. |
| B) Any class. |
| C) Any class in com.mycompany package. |
| D) Any class that extends Hotel. |

15

Given the code. What is the result?

1.      int i = 10;  
2.      **while** (++i <= 10) {  
3.          i++;  
4.      }  
5.      System.out.print(i);

|  |
| --- |
| A) 10 |
| B) 11 |
| C) 12 |
| D) Line 5 will be never reached. |

16

Given the code. What is the result?

String test = "This is a test string";  
String[] tokens = test.split("\\s");  
System.out.println(tokens.length);

|  |
| --- |
| A) 0 |
| B) 5 |
| C) 21 |
| D) Compilation fails |
| E) An exception is thrown at runtime |

17

Given the code. What is the result?

**public** **class** Test {   
    **private** **static** void doStuff(String str) {  
        int var = 4;  
        **if** (var == str.length()) {  
            System.out.print(str.charAt(var--) + " ");  
        }  
        **else** {  
            System.out.print(str.charAt(0) + " ");  
        }  
    }  
    **public** **static** void main(String args[]) {  
        doStuff("abcd");  
        doStuff("efg");  
        doStuff("hi");  
    }  
}

|  |
| --- |
| A) Compilation fails. |
| B) An exception is thrown at runtime. |
| C) d e h |
| D) d f i |
| E) c f i |
| F) c e h |

18

Given two classes defined in two different files. What is required at line marked "//some code goes here" to process the method doStuff() of a class A?

// The first file  
**package** pack1;  
  
**public** **class** ClassA {  
    **public** **static** void doStuff() {  
        System.out.println("doStuff");  
    }  
}  
// The second file  
**package** pack2;  
  
**public** **class** ClassB {  
    **public** **static** void main(String args[]) {  
        //some code goes here  
    }  
}

|  |
| --- |
| A) ClassA.doStuff(); |
| B) pack1.ClassA.doStuff(); |
| C) doStuff(); |
| D) It is impossible to use the method doStuff() in the class B. |
| E) import pack1.A.\*; doStuff(); |

19

A developer is creating a class A, that needs to access a class B. The B class is deployed in a jar named superLib.jar. Which three, will allow the developer to use the B class while compiling the A class. (choose three)

|  |
| --- |
| A) The jar file is located at /myfolder/superLib.jar and the A class is compiled using "javac -classpath /myfolder/superLib.jar A.java" |
| B) The jar file is located at /myfolder/superLib.jar and the A class is compiled using "javac -d /myfolder/superLib.jar A.java" |
| C) The jar file is located at /myfolder/superLib.jar and the A class is compiled using "javac -cp /myfolder/superLib.jar/B A.java" |
| D) The jar file is locate at /myfolder/superLib.jar and a classpath environment variable is set that includes /myfolder/superLib.jar |
| E) The jar file is located at /myfolder/superLib.jar and a classpath environment variable is set that includes /myfolder/superLib.jar/B.class |
| F) The jar file is located at $JAVA\_HOME/jre/lib/ext/superLib.jar |
| G) The jar file is located at $JAVA\_HOME/jre/classes/superLib.jar |

20

A Java bean component has the following field:   
private boolean done;  
Which method declarations follow the JavaBean standards for getting/settings this field? (Choose 3)

|  |
| --- |
| A) public void setDone(boolean done) |
| B) public boolean setDone(boolean done) |
| C) private boolean setDone(boolean done) |
| D) public void setDone() |
| E) public boolean getDone() |
| F) public boolean isDone() |
| G) public boolean getDone(boolean done) |
| H) public void isDone() |

21

Given the code. What is the result?

**import** java.util.HashSet;  
  
**public** **class** HashTest {  
      
    **private** String str;  
      
    **public** HashTest(String str) {  
        **this**.str = str;  
    }  
          
    @Override  
    **public** int hashCode() {               
        **return** **this**.str.hashCode();  
    }  
      
    @Override  
    **public** boolean equals(Object obj) {   
        **return** **this**.str.equals(obj);  
    }  
      
    **public** **static** void main(String args[]) {  
        HashTest h1 = **new** HashTest("1");  
        HashTest h2 = **new** HashTest("1");  
        String s1 = **new** String("2");  
        String s2 = **new** String("2");  
          
        HashSet<Object> hs = **new** HashSet<Object>();  
        hs.add(h1);  
        hs.add(h2);  
        hs.add(s1);  
        hs.add(s2);  
          
        System.out.print(hs.size());  
    }  
}

|  |
| --- |
| A) "4" is printed. |
| B) "3" is printed. |
| C) "2" is printed. |
| D) Compilation fails. |
| E) An exception is thrown at runtime. |

22

Given the code. What is the output?

**public** **class** Test {       
    int a = 10;  
      
    **public** void doStuff(int a) {  
        a += 1;  
        System.out.println(a++);  
    }  
    **public** **static** void main(String args[]) {  
        Test t = **new** Test();  
        t.doStuff(3);  
    }  
}

|  |
| --- |
| A) 11 |
| B) 12 |
| C) 4 |
| D) 5 |

23

Given the code. What is the output?

**public** **class** Hotel {  
    **private** int roomNr;  
      
    **public** Hotel(int roomNr) {  
        **this**.roomNr = roomNr;  
    }  
      
    **public** int getRoomNr() {  
        **return** **this**.roomNr;  
    }  
      
    **static** Hotel doStuff(Hotel hotel) {  
        hotel = **new** Hotel(1);  
        **return** hotel;  
    }  
      
    **public** **static** void main(String args[]) {  
        Hotel h1 = **new** Hotel(100);  
        System.out.print(h1.getRoomNr() + " ");  
        Hotel h2 = doStuff(h1);  
        System.out.print(h1.getRoomNr() + " ");  
        System.out.print(h2.getRoomNr() + " ");  
        h1 = doStuff(h2);  
        System.out.print(h1.getRoomNr() + " ");  
        System.out.print(h2.getRoomNr() + " ");  
    }  
}

|  |
| --- |
| A) 100 1 1 1 1 |
| B) 100 100 1 1 1 |
| C) 100 100 100 1 1 |
| D) 100 100 100 100 1 |
| E) 100 100 100 100 100 |

24

Given the code. What is the result?

**public** **class** Test {   
    **private** **static** void doStuff(String str) {  
        int var = 4;  
        **if** (var == str.length()) {  
            System.out.print(str.charAt(--var) + " ");  
        }  
        **else** {  
            System.out.print(str.charAt(0) + " ");  
        }  
    }  
    **public** **static** void main(String args[]) {  
        doStuff("abcd");  
        doStuff("efg");  
        doStuff("hi");  
    }  
}

|  |
| --- |
| A) Compilation fails. |
| B) An exception is thrown at runtime. |
| C) d e h |
| D) d f i |
| E) c f i |
| F) c e h |

25

What is true? (Choose three)

|  |
| --- |
| A) A method with the same signature as a private final method in class Z can be implemented in a subclass of Z. |
| B) A final method in class Z can be abstract if and only if Z is abstract. |
| C) A protected method in class Z can be overriden by any subclass of Z. |
| D) A private static method can be called only within other static methods in class Z. |
| E) A non-static public final method in class Z can be overriden in any subclass of Z. |
| F) A public static method in class Z can be called by a subclass of Z without explicitly referencing the class Z. |

26

What should be called after the code below to suggest that the JVM expend effort toward recycling the memory used by the object a? (Select two)

BigObject a = MyFactory.createBigObject();  
a.doStuff();  
a = null;

|  |
| --- |
| A) System.gc() |
| B) Runtime.gc() |
| C) System.freeMemory() |
| D) Runtime.getRuntime().freeMemory() |
| E) Runtime.getRuntime().gc() |

27

Given the code. What is the result?

**public** **class** SomeClass {  
    **private** int value = 1;  
      
    **public** int getValue() {  
        **return** value;  
    }  
      
    **public** void changeVal(int value) {  
        value = value;  
    }  
  
    **public** **static** void main(String args[]) {  
        int a = 2;  
        SomeClass c = **new** SomeClass();  
        c.changeVal(a);  
        System.out.print(c.getValue());  
    }  
}

|  |
| --- |
| A) "1" is printed |
| B) "2" is printed |
| C) Compilation fails |
| D) An exception is thrown at runtime |

28

What can directly access and change the value of the variable roomNr?

**package** com.mycompany;  
  
**public** **class** Hotel  {  
    **public** int roomNr = 100;  
}

|  |
| --- |
| A) Only the Hotel class. |
| B) Any class. |
| C) Any class in com.mycompany package. |
| D) Any class that extends Hotel. |

29

Given the code. What is the result?

**class** Hotel {  
    **public** int bookings;  
    **public** void book() {  
        bookings++;  
    }  
}  
  
**public** **class** SuperHotel **extends** Hotel {  
    **public** void book() {  
        bookings--;  
    }  
      
    **public** void book(int size) {  
        book();  
        **super**.book();  
        bookings += size;  
    }  
      
    **public** **static** void main(String args[]) {  
        Hotel hotel = **new** SuperHotel();  
        hotel.book(2);  
        System.out.print(hotel.bookings);  
    }  
}

|  |
| --- |
| A) Compilation fails. |
| B) An exception is thrown at runtime. |
| C) 0 |
| D) 1 |
| E) 2 |
| F) -1 |

30

Given the code. What is the result when this program is executed?

**public** **class** SuperHotel {  
    **static** int x[];  
      
    **static** {  
        x[0] = 1;  
    }  
      
    **public** **static** void main(String args[]) {          
    }     
}

|  |
| --- |
| A) ArrayIndexOutOfBoundsException is thrown |
| B) ExceptionInInitializerError is thrown |
| C) IllegalStateException is thrown |
| D) StackOverflowException is thrown |

31

Given the code. What is the result?

**public** **class** SomeClass {  
    **private** int value = 1;  
      
    **public** void printVal(int value) {  
        System.out.print(value);  
    }  
  
    **public** **static** void main(String args[]) {  
        int a = 2;  
        SomeClass c = **new** SomeClass();  
        c.printVal(a);  
    }  
}

|  |
| --- |
| A) "1" is printed |
| B) "2" is printed |
| C) Compilation fails |
| D) An exception is thrown at runtime |

32

Given the code. What is the result?

**public** **static** void main(String args[]) {  
        **try** {  
            String arr[] = **new** String[10];  
            arr = null;  
            arr[0] = "one";  
            System.out.print(arr[0]);  
        } **catch**(NullPointerException nex) {   
            System.out.print("null pointer exception");   
        } **catch**(Exception ex) {  
            System.out.print("exception");  
        }  
    }

|  |
| --- |
| A) "one" is printed. |
| B) "exception" is printed. |
| C) "null pointer exception" is printed. |
| D) Compilation fails. |

33

Given the code. What is the result?

**public** **static** void main(String args[]) {  
        **try** {  
            String arr[] = **new** String[10];  
            arr = null;  
            arr[0] = "one";  
            System.out.print(arr[0]);  
        } **catch**(Exception ex) {  
            System.out.print("exception");  
        } **catch**(NullPointerException nex) {  
            System.out.print("null pointer exception");  
        }     
    }

|  |
| --- |
| A) "one" is printed. |
| B) "exception" is printed. |
| C) "null pointer exception" is printed. |
| D) Compilation fails. |

34

Given the code. What is the result?

**class** Vehicle {  
    **public** void printSound() {  
        System.out.print("vehicle");  
    }  
}  
  
**class** Car **extends** Vehicle {  
    **public** void printSound() {  
        System.out.print("car");  
    }  
}  
  
**class** Bike **extends** Vehicle {  
    **public** void printSound() {  
        System.out.print("bike");  
    }  
}  
  
**public** **class** Test {  
    **public** **static** void main(String[] args) {  
        Vehicle v = **new** Car();  
        Car c = (Car) v;  
          
        v.printSound();  
        c.printSound();  
    }     
}

|  |
| --- |
| A) Compilation fails. |
| B) An exception is thrown at runtime. |
| C) "vehiclecar" is printed. |
| D) "vehiclebike" is printed. |
| E) "carcar" is printed. |
| F) "bikebike" is printed |

35

Given the code. What is true?

**public** **class** Room {  
    **public** int roomNr;  
    **private** Date beginDtm;  
    **private** Date endDttm;  
      
    **public** void book(int roomNr, Date beginDttm, Date endDttm) {  
        **this**.roomNr = roomNr;  
        **this**.beginDtm = beginDttm;  
        **this**.endDttm = endDttm;  
    }  
}

|  |
| --- |
| A) The code demonstrates polymorphism. |
| B) The class is fully encapsulated. |
| C) The variable roomNr breaks encapsulation. |
| D) Variables beginDttm and endDttm break polymorphism. |
| E) The method book breaks encapsulation. |

36

What is true about has-a and is-a relationships? (Choose two)

|  |
| --- |
| A) Instance variables can be used when creating a has-a relationship. |
| B) Inheritance represents an is-a relationship. |
| C) Inheritance represents a has-a relationship. |
| D) Instances must be used when creating a has-a relationship. |

37

Given the code. What is the result?

1   **public** **class** TryMe {  
2       Integer A;  
3       int a;  
4       **public** TryMe(int a) {  
5           **this**.a = A + a;  
6           System.out.print(**this**.a);  
7       }  
8       **public** **static** void main(String args[]) {          
9           Integer A = **new** Integer(1);  
10          TryMe t = **new** TryMe(A);  
11      }  
12  }

|  |
| --- |
| A) The value "1" is printed |
| B) Compilation fails because of an error in line 5 |
| C) A NullPointerException occurs at runtime |
| D) A NumberFormatException occurs at runtime |
| E) An IllegalStateExcepition occurs at runtime |

38

Given the code. What is the output?

1.  **public** **static** void main(String args[]) {  
2.      Object myObj = **new** String[]{"one", "two", "three"};{  
3.          **for** (String s : (String[])myObj) System.out.print(s + ".");  
4.      }  
5.  }

|  |
| --- |
| A) one.two.three. |
| B) Compilation fails because of an error at line 2 |
| C) Compilation fails because of an error at line 3 |
| D) An exception is thrown at runtime. |

39

Given the code. What is the result?

String test = "This is a test string";  
String[] tokens = test.split("\s");  
System.out.println(tokens.length);

|  |
| --- |
| A) 0 |
| B) 5 |
| C) 21 |
| D) Compilation fails |
| E) An exception is thrown at runtime |