TB9InheritanceMC Name

1. Consider the following two classes.

```
public class A
{
  public void show()
  {
    System.out.print("A");
  }
}


public class B extends A
{
  public void show()
  {
    System.out.print("B");
  }
}
```

What is printed as a result of executing the following code segment?

```
A obj = new B();
obj.show();
```

(A) A

(B) B

(C) AB

(D) BA

(E) The code results in a runtime error.

TB9InheritanceMC

2. Consider the following class definitions.

```
public class Robot
{
   private int servoCount;
   public int getServoCount()
   {
     return servoCount;
   }
   public void setServoCount(int in)
   {
     servoCount = in;
   }
}
public class Android extends Robot
{
   private int servoCount;
   public Android(int initVal)
   {
     setServoCount(initVal);
   }
   public int getServoCount()
   {
     return super.getServoCount();
   }
   public int getLocal()
   {
     return servoCount;
   }
   public void setServoCount(int in)
   {
     super.setServoCount(in);
   }
   public void setLocal(int in)
   {
     servoCount = in;
   }
}
```

The following code segment appears in a method in another class.

```
int x = 10;
int y = 20;
/* missing code */
```

Which of the following code segments can be used to replace `/* missing code */` so that the value `20` will be printed?

TB9InheritanceMC

(A)
```
Android a = new Android(x);
a.setServoCount(y);
System.out.println(a.getServoCount());
```

(B)
```
Android a = new Android(x);
a.setServoCount(y);
System.out.println(a.getLocal());
```

(C)
```
Android a = new Android(x);
a.setLocal(y);
System.out.println(a.getServoCount());
```

(D)
```
Android a = new Android(y);
a.setServoCount(x);
System.out.println(a.getLocal());
```

(E)
```
Android a = new Android(y);
a.setLocal(x);
System.out.println(a.getLocal());
```

3. Consider the following class definitions.

```
public class Artifact
{
   private String title;
   private int year;
   public Artifact(String t, int y)
   {
     title = t;
     year = y;
   }
   public void printInfo()
   {
     System.out.print(title + " (" + year + ")");
   }
}
public class Artwork extends Artifact
{
   private String artist;
   public Artwork(String t, int y, String a)
   {
     super(t, y);
     artist = a;
   }
   public void printInfo()
   {
     /* missing implementation */
   }
}
```

The following code segment appears in a method in another class.

```
Artwork starry = new Artwork("The Starry Night", 1889, "Van Gogh");
starry.printInfo();
```

The code segment is intended to produce the following output.

```
The Starry Night (1889) by Van Gogh
```

Which of the following can be used to replace `/* missing implementation */` in the `printInfo` method in the `Artwork` class so that the code segment produces the intended output?

TB9InheritanceMC

(A)
```
System.out.print(title + " (" + year + ") by " + artist);
```

(B)
```
super.printInfo(artist);
```

(C)
```
System.out.print(super.printInfo() + " by " + artist);
```

(D)
```
super();
System.out.print(" by " + artist);
```

(E)
```
super.printInfo();
System.out.print(" by " + artist);
```

4. Consider the following class definition.

```
public class Backyard
{
   private int length;
   private int width;
   public Backyard(int l, int w)
   {
     length = l;
     width = w;
   }
   public int getLength()
   {
     return length;
   }
   public int getWidth()
   {
     return width;
   }
   public boolean equals(Object other)
   {
     if (other == null)
     {
return false;
     }
     Backyard b = (Backyard) object;
     return (length == b.getLength() & width == b.getWidth());
   }
}
```

The following code segment appears in a class other than `Backyard`. It is intended to print `true` if `b1` and `b2` have the same lengths and widths, and to print `false` otherwise. Assume that `x`, `y`, `j`, and `k` are properly declared and initialized variables of type `int`.

```
Backyard b1 = new Backyard(x, y);
Backyard b2 = new Backyard(j, k);
System.out.println( /* missing code */ );
```

Which of the following can be used as a replacement for `/* missing code */` so the code segment works as intended?

TB9InheritanceMC

(A) `b1 == b2`

(B) `b1.equals(b2)`

(C) `equals(b1, b2)`

(D) `b1.equals(b2.getLength(), b2.getWidth())`

(E) `b1.length == b2.length && b1.width == b2.width`

5. Consider the following class definition.

```
public class Beverage
{
   private int temperature;
   public Beverage(int t)
   {
     temperature = t;
   }
   public int getTemperature()
   {
     return temperature;
   }
   public boolean equals(Object other)
   {
     if (other == null)
     {
return false;
     }
     Beverage b = (Beverage) other;
     return (b.getTemperature() == temperature);
   }
}
```

The following code segment appears in a class other than `Beverage`. Assume that `x` and `y` are properly declared and initialized `int` variables.

```
Beverage hotChocolate = new Beverage(x);
Beverage coffee = new Beverage(y);
boolean same = /* missing code */;
```

Which of the following can be used as a replacement for `/* missing code */` so that the `boolean` variable `same` is set to `true` if and only if the `hotChocolate` and `coffee` objects have the same temperature values?

TB9InheritanceMC

(A) `(hotChocolate = coffee)`

(B) `(hotChocolate == coffee)`

(C) `hotChocolate.equals(coffee)`

(D) `hotChocolate.equals(coffee.getTemperature())`

(E) `hotChocolate.getTemperature().equals(coffee.getTemperature())`

6. Consider the following Book and AudioBook classes.

```
public class Book
{
  private int numPages;
  private String bookTitle;

  public Book(int pages, String title)
  {
    numPages = pages;
    bookTitle = title;
  }

  public String toString()
  {
    return bookTitle + " " + numPages;
  }

  public int length()
  {
    return numPages;
  }
}


public class AudioBook extends Book
{
  private int numMinutes;

  public AudioBook(int minutes, int pages, String title)
  {
    super(pages, title);
    numMinutes = minutes;
  }

  public int length()
  {
    return numMinutes;
  }

  public double pagesPerMinute()
  {
    return ((double) super.length()) / numMinutes;
  }
}
```

Consider the following code segment that appears in a class other than Book or AudioBook.

```
Line 1:  Book[] books = new Book[2];
Line 2:  books[0] = new AudioBook(100, 300, "The Jungle");
Line 3:  books[1] = new Book(400, "Captains Courageous");
Line 4:  System.out.println(books[0].pagesPerMinute());
Line 5:  System.out.println(books[0].toString());
Line 6:  System.out.println(books[0].length());
Line 7:  System.out.println(books[1].toString());
```

Which of the following best explains why the code segment will not compile?

(A) Line 2 will not compile because variables of type Book may not refer to variables of type AudioBook.

(B) Line 4 will not compile because variables of type Book may only call methods in the Book class.

(C) Line 5 will not compile because the AudioBook class does not have a method named toString declared or implemented.

(D) Line 6 will not compile because the statement is ambiguous. The compiler cannot determine which length method should be called.

(E) Line 7 will not compile because the element at index 1 in the array named books may not have been initialized.

7. Consider the following class definition.

```
public class Document
{
   private int pageCount;
   private int chapterCount;
   public Document(int p, int c)
   {
     pageCount = p;
     chapterCount = c;
   }
   public String toString()
   {
     return pageCount + " " + chapterCount;
   }
}
```

The following code segment, which is intended to print the page and chapter counts of a `Document` object, appears in a class other than `Document`.

```
Document d = new Document(245, 16);
System.out.println( /* missing code */ );
```

Which of the following can be used as a replacement for `/* missing code */` so the code segment works as intended?

(A) `d.toString()`

(B) `toString(d)`

(C) `d.pageCount + " " + d.chapterCount`

(D) `d.getPageCount() + " " + d.getChapterCount()`

(E) `Document.pageCount + " " + Document.chapterCount`

8. A car dealership needs a program to store information about the cars for sale. For each car, they want to keep track of the following information: number of doors (2 or 4), whether the car has air conditioning, and its average number of miles per gallon. Which of the following is the best object-oriented program design?

(A) Use one class, Car, with three instance variables:
int numDoors, boolean hasAir, and
double milesPerGallon.

(B) Use four unrelated classes: Car, Doors, AirConditioning, and MilesPerGallon.

(C) Use a class Car with three subclasses: Doors, AirConditioning, and MilesPerGallon.

(D) Use a class Car, with a subclass Doors, with a subclass
AirConditioning, with a subclass MilesPerGallon.

(E) Use three classes: Doors, AirConditioning, and MilesPerGallon, each with a subclass Car.

TB9InheritanceMC

9. Consider the following class definitions.

```
public class ClassA
{
   public String getValue()
   {
     return "A";
   }
   public void showValue()
   {
     System.out.print(getValue());
   }
}
public class ClassB extends ClassA
{
   public String getValue()
   {
     return "B";
   }
}
```

The following code segment appears in a class other than `ClassA` or `ClassB`.

```
ClassA obj = new ClassB();
obj.showValue();
```

What, if anything, is printed when the code segment is executed?

(A) `A`

(B) `B`

(C) `AB`

(D) `BA`

(E) Nothing is printed because the code does not compile.

TB9InheritanceMC

10. Consider the following class declarations.

```
public class A
{
private int x;
public A()
{ x = 0; }
public A(int y)
{ x = y; }
// There may be instance variables, constructors, and methods that are not shown.
}

public class B extends A
{
private int y;
public B()
{
/* missing code */
}
// There may be instance variables, constructors, and methods that are not shown.
}
```

Which of the following can be used to replace /* *missing code* */ so that the statement

```
B temp = new B();
```

will construct an object of type B and initialize both x and y with 0 ?

1. y = 0
2. super (0); y = 0;
3. x = 0; y = 0;

(A) I only

(B) II only

(C) I and II only

(D) II and III only

(E) I, II, and III

11. Consider the following class definitions.

```
public class Book
{
   private String bookTitle;
   public Book()
   {
     bookTitle = "";
   }
   public Book(String title)
   {
     bookTitle = title;
   }
}
public class TextBook extends Book
{
   private String subject;
   public TextBook(String theSubject)
   {
     subject = theSubject;
   }
}
```

The following code segment appears in a method in a class other than `Book` or `TextBook`.

```
Book b = new TextBook("Psychology");
```

Which of the following best describes the effect of executing the code segment?

TB9InheritanceMC

(A) The `TextBook` constructor initializes the instance variable `subject` with the value of the parameter `theSubject`, and then invokes the zero-parameter `Book` constructor, which initializes the instance variable `bookTitle` to `""`.

(B) The `TextBook` constructor initializes the instance variable `subject` with the value of the parameter `theSubject`, and then invokes the one-parameter `Book` constructor with `theSubject` as the parameter, which initializes the instance variable `bookTitle` to the value of the parameter `theSubject`.

(C) There is an implicit call to the zero-parameter `Book` constructor. The instance variable `bookTitle` is then initialized to `""`. Then, the instance variable `subject` is initialized with the value of the parameter `theSubject`.

(D) The code segment will not execute because the `TextBook` constructor does not contain an explicit call to one of the `Book` constructors.

(E) The code segment will not execute because the `TextBook` constructor does not have a parameter for the title of the book.

TB9InheritanceMC

12. Consider the following two classes.

```
public class Dog
{
  public void act()
  {
    System.out.print("run ");
    eat();
  }
  public void eat()
  {
    System.out.print("eat ");
  }
}
public class UnderDog extends Dog
{
  public void act()
  {
    super.act();
    System.out.print("sleep ");
  }
  public void eat()
  {
    super.eat();
    System.out.print("bark ");
  }
}
```

Assume that the following declaration appears in a class other than Dog.

Dog fido = new UnderDog ( ) ;

What is printed as a result of the call fido.act() ?

(A) run eat

(B) run eat sleep

(C) run eat sleep bark

(D) run eat bark sleep

(E) Nothing is printed due to infinite recursion.

13. Consider the following class definitions.

```
public class Bike
{
   private int numWheels = 2;
   // No constructor defined
}
public class EBike extends Bike
{
   private int numBatteries;
   public EBike(int batteries)
   {
     numBatteries = batteries;
   }
}
```

The following code segment appears in a method in a class other than `Bike` or `EBike`.

```
EBike eB = new EBike(4);
```

Which of the following best describes the effect of executing the code segment?

TB9InheritanceMC

(A) An implicit call to the zero-parameter `Bike` constructor initializes the instance variable `numWheels`. The instance variable `numBatteries` is initialized using the value of the parameter `batteries`.

(B) An implicit call to the one-parameter `Bike` constructor with the parameter passed to the `EBike` constructor initializes the instance variable `numWheels`. The instance variable `numBatteries` is initialized using the value of the parameter `batteries`.

(C) Because `super` is not explicitly called from the `EBike` constructor, the instance variable `numWheels` is not initialized. The instance variable `numBatteries` is initialized using the value of the parameter `batteries`.

(D) The code segment will not execute because the `Bike` class is a superclass and must have a constructor.

(E) The code segment will not execute because the constructor of the `EBike` class is missing a second parameter to use to initialize the `numWheels` instance variable.

TB9InheritanceMC

14. Consider the following class definitions.

```
public class Computer
{
   private String memory;
   public Computer()
   {
     memory = "RAM";
   }
   public Computer(String m)
   {
     memory = m;
   }
   public String getMemory()
   {
     return memory;
   }
}
public class Smartphone extends Computer
{
   private double screenWidth, screenHeight;
   public SmartPhone(double w, double h)
   {
     super("flash");
     screenWidth = w;
     screenHeight = h;
   }
   public double getScreenWidth()
   {
     return screenWidth;
   }
   public double getScreenHeight()
   {
     return screenHeight;
   }
}
```

The following code segment appears in a class other than `Computer` or `Smartphone`.

```
Computer myPhone = new SmartPhone(2.55, 4.53);
System.out.println("Device has memory: " + myPhone.getMemory() +
   ", screen area: " + myPhone.getScreenWidth() *
   myPhone.getScreenHeight() + " square inches.");
```

The code segment is intended to produce the following output.

```
Device has memory: flash, screen area: 11.5515 square inches.
```

Which of the following best explains why the code segment does not work as intended?

(A) An error occurs during compilation because a `Smartphone` object cannot be assigned to the `Computer` reference variable `myPhone`.

(B) An error occurs during compilation because the `Smartphone` class has no `getMemory` method.

(C) An error occurs during compilation because the `getScreenWidth` and `getScreenHeight` methods are not defined for the `Computer` object `myPhone`.

(D) An error occurs at runtime because the `Smartphone` class has no `getMemory` method.

(E) An error occurs at runtime because the `getScreenWidth` and `getScreenHeight` methods are not defined for the `Computer` object `myPhone`.

15. Consider the following class definitions.

```
public class C1
{
   public C1()
   { /* implementation not shown */ }
   public void m1()
   { System.out.print("A"); }
   public void m2()
   { System.out.print("B"); }
}
public class C2 extends C1
{
   public C2()
   { /* implementation not shown */ }
   public void m2()
   { System.out.print("C"); }
}
```

The following code segment appears in a class other than `C1` or `C2`.

```
C1 obj1 = new C2();
obj1.m1();
obj1.m2();
```

The code segment is intended to produce the output `AB`. Which of the following best explains why the code segment does not produce the intended output?

TB9InheritanceMC

(A) A compile-time error occurs because `obj1` is declared as type `C1` but instantiated as type `C2`.

(B) A runtime error occurs because method `m1` does not appear in `C2`.

(C) Method `m1` is not executed because it does not appear in `C2`.

(D) Method `m2` is executed from the subclass instead of the superclass because `obj1` is instantiated as a `C2` object.

(E) Method `m2` is executed twice (once in the subclass and once in the superclass) because it appears in both classes.

TB9InheritanceMC

16. Consider the following two class definitions.

```
public class Bike
{
   private int numOfWheels = 2;
   public int getNumOfWheels()
   {
     return numOfWheels;
   }
}
public class EBike extends Bike
{
   private int numOfWatts;
   public EBike(int watts)
   {
     numOfWatts = watts;
   }
   public int getNumOfWatts()
   {
     return numOfWatts;
   }
}
```

The following code segment occurs in a class other than `Bike` or `EBike`.

```
Bike b = new EBike(250);
System.out.println(b.getNumOfWatts());
System.out.println(b.getNumOfWheels());
```

Which of the following best explains why the code segment does not compile?

(A) The `Bike` superclass does not have a constructor.

(B) There are too many arguments to the `EBike` constructor call in the code segment.

(C) The first line of the subclass constructor is not a call to the superclass constructor.

(D) The `getNumOfWatts` method is not found in the `Bike` class.

(E) The `getNumOfWheels` method is not found in the `EBike` class.

17. Consider the following class definitions.

```
public class Game
{
   private String name;
   public Game(String n)
   {
     name = n;
   }
   // Rest of definition not shown
}
public class BoardGame extends Game
{
   public BoardGame(String n)
   {
     super(n);
   }
   // Rest of definition not shown
}
```

The following code segment appears in a class other than `Game` or `BoardGame`.

```
Game g1 = new BoardGame("checkers");
BoardGame g2 = new Game("chess");
ArrayList<Game> My_Games = new ArrayList();
My_Games.add(g1);
My_Games.add(g2);
```

Which of the following best explains why the code segment does not compile?

(A) A `BoardGame` object cannot be assigned to the `Game` reference `g1`.

(B) A `Game` object cannot be assigned to the `BoardGame` reference `g2`.

(C) The `My_Games` object cannot contain elements of different types.

(D) The object referenced by `g1` cannot be added to `My_Games` since `g1` was instantiated by a call to the `BoardGame` constructor.

(E) The object referenced by `g2` cannot be added to `My_Games` since `g2` was declared to be of type `BoardGame`.

18. Consider the following class definitions.

```
public class Bird
{
   private int beakStrength;
   public void Bird(int input)
   {
     beakStrength = input;
   }
   public void setBeakStrength(int strength)
   {
     beakStrength = strength;
   }
}
public class Hawk extends Bird
{
   private int talonStrength;
   public Hawk(int talon, int beak)
   {
     super(beak);
     talonStrength = talon;
   }
}
```

The following statement appears in a method in another class.

```
Bird b = new Hawk(5, 8);
```

Which of the following best describes the effect of executing the statement?

TB9InheritanceMC

(A) The `Bird` variable `b` is instantiated as a `Hawk`. The instance variable `talonStrength` is initialized with the value from the parameter `talon`. The `Hawk` constructor cannot set the instance variable `beakStrength` because a subclass does not have access to a private variable in its superclass.

(B) The `Bird` variable `b` is instantiated as a `Hawk`. The call `super(beak)` returns a value from the instance variable `beakStrength` in the superclass and makes it accessible in the subclass. The instance variable `talonStrength` is then initialized with the value from the parameter `talon`.

(C) The `Bird` variable `b` is instantiated as a `Hawk`. The instance variable `talonStrength` is initialized with the value from the parameter `talon`. No other initializations are made to any instance variables.

(D) The `Bird` variable `b` is instantiated as a `Hawk`. The call `super(beak)` invokes the `Bird` constructor and initializes the instance variable `beakStrength` with the value from the parameter `beak`. The instance variable `talonStrength` is then initialized with the value from the parameter `talon`.

(E) The code segment will not execute because the `Bird` variable `b` cannot be instantiated as a `Hawk`.

TB9InheritanceMC

19. Consider the following classes.

```
public class Base
{
public Base()
{
System.out.print("Base" + " ");
}
}
public class Derived extends Base
{
public Derived()
{
System.out.print("Derived" + " ");
}
}
```

Assume that the following statement appears in another class.

```
Derived d1 = new Derived();
```

What is printed as a result of executing the statement?

(A) Nothing is printed because the statement is a variable declaration.

(B) Base

(C) Derived

(D) Base Derived

(E) Derived Base

20. Consider the following partial class definitions.

```
public class Membership
{
   private String id;
   public Membership(String input)
   { id = input; }
   // Rest of definition not shown
}
public class FamilyMembership extends Membership
{
   private int numberInFamily = 2;
   public FamilyMembership(String input)
   { super(input); }
   public FamilyMembership(String input, int n)
   {
     super(input);
     numberInFamily = n;
   }
   // Rest of definition not shown
}
public class IndividualMembership extends Membership
{
   public IndividualMembership(String input)
   { super(input); }
   // Rest of definition not shown
}
```

The following code segment occurs in a class other than `Membership`, `FamilyMembership`, or `IndividualMembership`.

```
FamilyMembership m1 = new Membership("123"); // Line 1
Membership m2 = new IndividualMembership("456"); // Line 2
Membership m3 = new FamilyMembership("789"); // Line 3
FamilyMembership m4 = new FamilyMembership("987", 3); // Line 4
Membership m5 = new Membership("374"); // Line 5
```

Which of the following best explains why the code segment does not compile?

TB9InheritanceMC

(A) In line 1, `m1` cannot be declared as type `FamilyMembership` and instantiated as a `Membership` object.

(B) In line 2, `m2` cannot be declared as type `Membership` and instantiated as an `IndividualMembership` object.

(C) In line 3, `m3` cannot be declared as type `Membership` and instantiated as a `FamilyMembership` object.

(D) In line 4, `m4` cannot be declared as type `FamilyMembership` and instantiated as a `FamilyMembership` object.

(E) In line 5, `m5` cannot be declared as type `Membership` and instantiated as a `Membership` object.

TB9InheritanceMC

21. Consider the following declaration for a class that will be used to represent points in the *xy*-coordinate plane.

```
public class Point
{
  private int x;      // x-coordinate of the point
  private int y;      // y-coordinate of the point

  public Point()
  {
    x = 0;
    y = 0;
  }

  public Point(int a, int b)
  {
    x = a;
    y = b;
  }

  // Other methods not shown
}
```

The following incomplete class declaration is intended to extend the above class so that points can be named.

```
public class NamedPoint extends Point
{
  private String name; // name of point

  // Constructors go here

  // Other methods not shown
}
```

Consider the following proposed constructors for this class.

I.
```
public NamedPoint()
{
  name = "";
}
```

II.
```
public NamedPoint(int d1, int d2, String pointName)
{
  x = d1;
  y = d2;
  name = pointName;
}
```

III.
```
public NamedPoint(int d1, int d2, String pointName)
{
  super(d1, d2);
  name = pointName;
}
```

Which of these constructors would be legal for the NamedPoint class?

TB9InheritanceMC

(A) I only

(B) II only

(C) III only

(D) I and III only

(E) II and III only

22. Consider the following class definitions.

```
public class Apple
{
   public void printColor()
   {
     System.out.print("Red");
   }
}
public class GrannySmith extends Apple
{
   public void printColor()
   {
     System.out.print("Green");
   }
}
public class Jonagold extends Apple
{
   // no methods defined
}
```

The following statement appears in a method in another class.

```
someApple.printColor();
```

Under which of the following conditions will the statement print `"Red"` ?

1. When `someApple` is an object of type `Apple`
2. When `someApple` is an object of type `GrannySmith`
3. When `someApple` is an object of type `Jonagold`

(A) I only

(B) I and II only

(C) I and III only

(D) II and III only

(E) I, II, and III

23. Consider the following class definitions.

```
public class Rectangle
{
   private int height;
   private int width;
   public Rectangle()
   {
     height = 1;
     width = 1;
   }
   public Rectangle(int x)
   {
     height = x;
     width = x;
   }
   public Rectangle(int h, int w)
   {
     height = h;
     width = w;
   }
   // There may be methods that are not shown.
}
public class Square extends Rectangle
{
   public Square(int x)
   {
     /* missing code */
   }
}
```

Which of the following code segments can replace `/* missing code */` so that the `Square` class constructor initializes the `Rectangle` class instance variables `height` and `width` to `x` ?

(A) `super();`

(B) `super(x);`

(C) `Rectangle(x);`

(D) `Square(x, x);`

(E)
```
height = x;
width = x;
```

24. Consider the following class definitions.

```
public class A
{
   public String message(int i)
   {
     return "A" + i;
   }
}
public class B extends A
{
   public String message(int i)
   {
     return "B" + i;
   }
}
```

The following code segment appears in a class other than `A` or `B`.

```
A obj1 = new B(); // Line 1
B obj2 = new B(); // Line 2
System.out.println(obj1.message(3));   // Line 3
System.out.println(obj2.message(2));   // Line 4
```

Which of the following best explains the difference, if any, in the behavior of the code segment that will result from removing the `message` method from class `A` ?

(A) The statement in line 3 will cause a compiler error because the `message` method for `obj1` cannot be found.

(B) The statement in line 4 will cause a compiler error because the `message` method for `obj2` cannot be found.

(C) As a result of the method call in line 3, the `message` method in class `B` will be executed instead of the `message` method in class `A`.

(D) As a result of the method call in line 4, the `message` method in class `B` will be executed instead of the `message` method in class `A`.

(E) The behavior of the code segment will remain unchanged.

25. Consider the following class definitions.

```
public class Road
{
   private String roadName;
   public Road(String name)
   {
     roadName = name;
   }
}
public class Highway extends Road
{
   private int speedLimit;
   public Highway(String name, int limit)
   {
     super(name);
     speedLimit = limit;
   }
}
```

The following code segment appears in a method in another class.

```
Road r1 = new Highway("Interstate 101", 55); // line 1
Road r2 = new Road("Elm Street"); // line 2
Highway r3 = new Road("Sullivan Street"); // line 3
Highway r4 = new Highway("New Jersey Turnpike", 65); // line 4
```

Which of the following best explains the error, if any, in the code segment?

TB9InheritanceMC

(A) Line 1 will cause an error because a `Road` variable cannot be instantiated as an object of type `Highway`.

(B) Line 2 will cause an error because the `Road` constructor is not properly called.

(C) Line 3 will cause an error because a `Highway` variable cannot be instantiated as an object of type `Road`.

(D) Line 4 will cause an error because the `Highway` constructor is not properly called.

(E) The code segment compiles and runs without error.

TB9InheritanceMC

26. Consider the following class definitions.

```
public class Pet
{
   public void speak()
   {
     System.out.print("pet sound");
   }
}
public class Dog extends Pet
{
   public void bark()
   {
     System.out.print("woof woof");
   }
   public void speak()
   {
     bark();
   }
}
public class Cat extends Pet
{
   public void speak()
   {
     System.out.print("meow meow");
   }
}
```

The following statement appears in a method in another class.

```
myPet.speak();
```

Under which of the following conditions will the statement compile and run without error?

1. When `myPet` is an object of type `Pet`
2. When `myPet` is an object of type `Dog`
3. When `myPet` is an object of type `Cat`

(A) I only

(B) I and II only

(C) I and III only

(D) II and III only

(E) I, II, and III

27. Consider the following interface and class declarations.

```
public interface Student
{  /* implementation not shown */  }

public class Athlete
{  /* implementation not shown */  }

public class TennisPlayer extends Athlete implements Student
{  /* implementation not shown */  }
```

Assume that each class has a zero-parameter constructor. Which of the following is NOT a valid declaration?

(A) Student a = new TennisPlayer();

(B) TennisPlayer b = new TennisPlayer();

(C) Athlete c = new TennisPlayer();

(D) Student d = new Athlete();

(E) Athlete e = new Athlete();

TB9InheritanceMC

28. Consider the following class definitions.

```
public class Hero
{
   private String name;
   private int power;
   public Hero(String n, int p)
   {
     name = n;
     power = p;
   }
   public void powerUp(int p)
   {
     power += p;
   }
   public int showPower()
   { return power; }
}
public class SuperHero extends Hero
{
   public SuperHero(String n, int p)
   {
     super(n, p);
   }
   public void powerUp(int p)
   {
     super.powerUp(p * 2);
   }
}
```

The following code segment appears in a class other than `Hero` and `SuperHero`.

```
Hero j = new SuperHero("JavaHero", 50);
j.powerUp(10);
System.out.println(j.showPower());
```

What is printed as a result of executing the code segment?

(A) 10

(B) 20

(C) 60

(D) 70

(E) 100

29. Consider the following class definitions.

```
public class Drink
{
   // implementation not shown
}
public class Coffee extends Drink
{
    // There may be instance variables and constructors that are not
shown.
   // No methods are defined for this class.
}
```

The following code segment appears in a method in a class other than `Drink` or `Coffee`.

```
Coffee myCup = new Coffee();
myCup.setSize("large");
```

Which of the following must be true so that the code segment will compile without error?

(A) The `Drink` class must have a public method named `getSize` that takes a `String` value as its parameter.

(B) The `Drink` class must have a public method named `getSize` that takes no parameters.

(C) The `Drink` class must have a public method named `setSize` that takes a `String` value as its parameter.

(D) The `Drink` class must have a public method named `setSize` that takes no parameters.

(E) The `Drink` class must have a `String` instance variable named `size`.

30. Consider the following class definitions.

```
public class Book
{
   private String author;
   private String title;
   public Book(String the_author, String the_title)
   {
     author = the_author;
     title = the_title;
   }
}
public class Textbook extends Book
{
   private String subject;
    public Textbook(String the_author, String the_title, String
the_subject)
   {
     /* missing implementation */
   }
}
```

Which of the following can be used to replace `/* missing implementation */` so that the `Textbook` constructor compiles without error?

TB9InheritanceMC

(A)
```
author = the_author;
title = the_title;
subject = the_subject;
```

(B)
```
super(the_author, the_title);
super(the_subject);
```

(C)
```
subject = the_subject;
super(the_author, the_title);
```

(D)
```
super(the_author, the_title);
subject = the_subject;
```

(E)
```
super(the_author, the_title, the_subject);
```

The following questions refer to the following classes:

```
public class First
{
    public String name()
    {
        return "First";
    }
}

public class Second extends First
{
    public void whoRules()
    {
```

TB9InheritanceMC

```
                System.out.print(super.name() + " rules");
                System.out.println(" but " + name() + " is even better");
        }
        public String name()
        {
                return "Second";
        }
}

public class Third extends Second
{
        public String name()
        {
                return "Third";
        }
}
```

TB9InheritanceMC

31. Consider the following code segment.

```
/* SomeType1 */ varA = new Second();
/* SomeType2 */ varB = new Third();

varA.whoRules();
varB.whoRules();
```

Which of the following could be used to replace /* *SomeType1* */ and /* *SomeType2* */ so that the code segment will compile without error?

	/* *SomeType1* */	/* *SomeType2* */
I.	First	Third
II.	Second	Second
III.	Third	Third

(A) I only

(B) II only

(C) III only

(D) I and II

(E) II and III

---

32. Consider the following code segment.

```
Second varSecond = new Second();
Third varThird = new Third();

varSecond.whoRules();
varThird.whoRules();
```

What is printed as a result of executing the code segment?

TB9InheritanceMC

A
First rules but Second is even better
First rules but Second is even better

B
First rules but Second is even better
First rules but Third is even better

C
First rules but Second is even better
Second rules but Second is even better

D
First rules but Second is even better
Second rules but Third is even better

E
Second rules but Second is even better
Second rules but Second is even better