

*Objective 1 - "Is-A" and Has-A" Class Interaction*

03. The *has-a* relationship describes

- (A) inheritance.
- (B) encapsulation.
- (C) polymorphism.
- (D) composition.

04. The *is-a* relationship describes

- (A) inheritance.
- (B) encapsulation.
- (C) polymorphism.
- (D) composition.

05. A class, which can use all the features of an established superclass, is

- (A) a static class.
- (B) a superclass.
- (C) a subclass.
- (D) overloaded.

06. An established class, whose members can all be used by a newly declared class, is

- (A) a static class.
- (B) a superclass.
- (C) a subclass.
- (D) overloaded.

08. A *truck*, which is a special *car* converted for off-roading with special shocks, mud tires and four-wheel drive is an example of

- (A) a superclass.
- (B) inheritance.
- (C) instantiation.
- (D) composition.

### Objective 3 - Accessing Inheritance Members

13. Consider the following class heading.

```
public class Person extends Student
```

What is not true about the class interaction of that class heading?

- (A) It indicates an "is-a" class interaction between the two classes.
- (B) It indicates an inheritance relationship between **Person** and **Student**
- (C) It indicates that **Person** is the superclass and **Student** is the subclass.
- (D) It indicates that **Student** is the superclass and **Person** is the subclass.

14. **Consider the following program for questions 14 and 15.**

```
public class Question1415{  
    public static void main(String args[]){  
        Student tom = new Student();  
        System.out.println("tom's age is " + tom.getAge());  
        System.out.println("tom's grade is " + tom.getGrade());  
    }  
}  
  
class Person{  
    private int age;  
  
    public int getAge()  
    {  
        return age;  
    }  
}  
  
class Student extends Person{  
    private int grade;  
  
    public int getGrade() {  
        return grade;  
    }  
}
```

This program compiles and executes without error or logic problems.  
What evidence exists that proves that inheritance is functional in this program?

- (A) The **Student** class extends the **Person** class.
- (B) The **tom** object has access to the **getGrade** method.
- (C) The **tom** object has access to the **getAge** method.
- (D) There is evidence of class interaction with composition, but not with inheritance.

15. What is the consequence of removing **extends Person** from the program above?

- (A) The class interaction will change from inheritance to composition.
- (B) The class interaction will change from composition to inheritance.
- (C) The program will compile, but it will not execute correctly.
- (D) There will no longer be any interaction between the **Person** class and the **Student** class.

16. Which of the following is not possible between classes that have an inheritance relationship?

- (A) Access from superclass to any subclass members
- (B) Access from subclass to superclass members
- (C) Access from subclass methods to subclass data attributes
- (D) Access from superclass methods to superclass data attributes

## Use this program segment for questions 17 & 18.

```
public class Demo
{
    public static void main(String args[])
    {
        Student tom = new Student(12);
        tom.showData();
    }
}

class Person
{
    public int age;

    public Person()
    {
        System.out.println("Person Parameter Constructor");
        age = 17;
    }

    public int getAge()    { return age; }
}

class Student extends Person
{
    public int grade;

    public Student(int g)
    {
        grade = g;
        System.out.println("Student Parameter Constructor");
    }

    public int getGrade()    { return grade; }

    public void showData()
    {
        System.out.println("Student's Grade is " + grade);
        System.out.println("Student's Age is " + age);
    }
}
```

17. What are the first 2 lines of output?

- (A) Person Parameter Constructor  
Student Parameter Constructor
- (B) Student Parameter Constructor  
Person Parameter Constructor
- (C) Person Parameter Constructor  
Person Parameter Constructor
- (D) Student Parameter Constructor  
Student Parameter Constructor
- (E) No Output.  
This program does not compile.

18. What are the last 2 lines of output?

- (A) Student's Grade is 12  
Student's Age is 17
- (B) Student's Grade is 12  
Student's Age is 17
- (C) Student's Grade is 12  
Student's Age is 17
- (D) Student's Grade is 12  
Student's Age is 17
- (E) No Output.  
This program does not compile.

## Use this program segment for questions 19 & 20.

```
public class Demo
{
    public static void main(String args[])
    {
        Student tom = new Student(12);
        tom.showData();
    }
}

class Person
{
    private int age;

    public Person()
    {
        System.out.println("Person Parameter Constructor");
        age = 17;
    }

    public int getAge()    { return age; }
}

class Student extends Person
{
    private int grade;

    public Student(int g)
    {
        grade = g;
        System.out.println("Student Parameter Constructor");
    }

    public int getGrade()    { return grade; }

    public void showData()
    {
        System.out.println("Student's Grade is " + grade);
        System.out.println("Student's Age is " + age);
    }
}
```

19. What are the first 2 lines of output?

- (A) Person Parameter Constructor  
Student Parameter Constructor
- (B) Student Parameter Constructor  
Person Parameter Constructor
- (C) Person Parameter Constructor  
Person Parameter Constructor
- (D) Student Parameter Constructor  
Student Parameter Constructor
- (E) No Output.  
This program does not compile.

20. What are the last 2 lines of output?

- (A) Student's Grade is 12  
Student's Age is 17
- (B) Student's Grade is 17  
Student's Age is 12
- (C) Student's Grade is 17  
Student's Age is 17
- (D) Student's Grade is 12  
Student's Age is 12
- (E) No Output.  
This program does not compile.

## Use this program segment for questions 21 & 22.

## Use this program segment for questions 23 & 24.

```
public class Demo
{
    public static void main(String args[])
    {
        Student tom = new Student(12,17);
        tom.showData();
    }
}

class Person
{
    private int age;

    public Person(int a)
    {
        System.out.println("Person Parameter Constructor");
        age = a;
    }

    public int getAge()    { return age; }
}

class Student extends Person
{
    private int grade;

    public Student(int a, int g)
    {
        super(a);
        grade = g;
        System.out.println("Student Parameter Constructor");
    }

    public int getGrade()    { return grade; }

    public void showData()
    {
        System.out.println("Student's Grade is " + getGrade());
        System.out.println("Student's Age is " + getAge());
    }
}
```

23. What are the first 2 lines of output?

- (A) Person Parameter Constructor  
Student Parameter Constructor
- (B) Student Parameter Constructor  
Person Parameter Constructor
- (C) Person Parameter Constructor  
Person Parameter Constructor
- (D) Student Parameter Constructor  
Student Parameter Constructor
- (E) No Output.  
This program does not compile.

24. What are the last 2 lines of output?

- (A) Student's Grade is 12  
Student's Age is 17
- (B) Student's Grade is 17  
Student's Age is 12
- (C) Student's Grade is 17  
Student's Age is 17
- (D) Student's Grade is 12  
Student's Age is 12
- (E) No Output.  
This program does not compile.

--	--

*Objective 4 - Inheritance Constructor Issues*

--

26. If the **super** keyword is used, in a constructor, to send information, where must it be placed?

- (A) Anywhere in the program
- (B) Anywhere in the subclass
- (C) Anywhere in the superclass
- (D) Anywhere in the superclass constructor
- (E) At the very beginning of the subclass constructor

27. How is information passed from the subclass constructor to the superclass constructor?

- (A) The superclass constructor is automatically called before the subclass constructor.
- (B) Use the **super** keyword followed by a parameter list for the superclass constructor.
- (C) Use the **super** keyword followed by the superclass identifier.
- (D) Use the **new** operator inside the subclass constructor to instantiate the superclass.

28. Consider the following class declaration.

```
public class Qwerty extends Widget
{
    private int count;

    public Qwerty(int c)
    {
        count = c;
    }
}
```

Which of the following **Qwerty** methods is identical to the one above?

<p>(A)</p> <pre>public Qwerty(int c) {     super(c);     count = c; }</pre>	<p>(B)</p> <pre>public Qwerty(int c) {     super();     count = c; }</pre>
<p>(C)</p> <pre>public Qwerty(int c) {     super(Widget);     count = c; }</pre>	<p>(D)</p> <pre>public Qwerty(int c) {     count = c;     super(); }</pre>

29. Consider the program segment and class declarations.

```
int widgetCount = 10;
int pidgetCount = 20;
Widget widget = new Pidget(widgetCount, pidgetCount);
```

```
public Widget
{
    private int numWidgets;

    public Widget(int nW)
    {
        numWidgets = nW;
    }
}
```



```
public class Pidget extends Widget
{
    private int numPidgets;
}
```

Which of the following **Pidget** constructors correctly initializes the instances variables?

(A)

```
public Pidget(int nW, int nP)
{
    numWidgets = nW;
    numPidgets = nP;
}
```

(B)

```
public Pidget(int nW, int nP)
{
    super(nw,nP);
}
```

(C)

```
public Pidget(int nW, int nP)
{
    super(nW);
    numPidgets = nP;
}
```

(D)

```
public Pidget(int nW, int nP)
{
    numPidgets = nP;
    super(nw);
}
```

30. Consider the program segment and class declarations.

```
int pidgetCount = 20;
Widget widget = new Widget(pidgetCount);
```

```
public Widget
{
    private int numWidgets;

    public Widget()
    {
        numWidgets = 0;
    }
}
```

```
public class Pidget extends Widget
{
    private int numPidgets;
}
```

Which of the following **Pidget** constructors correctly initializes the instances variables?

(A)

```
public Pidget(int nP)
{
    numWidgets = 0;
    numPidgets = nP;
}
```

(B)

```
public Pidget(int nP)
{
    super();
    numPidgets = nP;
}
```

(C)

```
public Pidget(int nP)
{
    super(nP);
}
```

(D)

```
public Pidget(int nP)
{
    numPidgets = nP;
    super();
}
```

31. Consider the program segment and class declarations.

```
int widgetCount = 10;
double widgetCost = 3.75;
int pidgetCount = 20;
int pidgetCost = 6.25;
Widget widget = new Pidget(widgetCount,widgetCost,pidgetCount,pidgetCost);

public Widget
{
    private int widgetCount;
    private double widgetCost;

    public Widget(int count, double cost)
    {
```

```

        widgetCount = count;
        widgetCost = cost;
    }
}

public class Pidget extends Widget
{
    private int pidgetCount;
    private double pidgetCost;
}

```

Which of the following **Pidget** constructors correctly initializes the instances variables?

(A)

```

public Pidget(int w1, double w2, int p1, double p2)
{
    super(w1,w2);
    pidgetCount = p1;
    pidgetCost = p2;
}

```

(B)

```

public Pidget(int w1, double w2, int p1, double p2)
{
    super(p1,p2);
    widgetCount = w1;
    widgetCost = w2;
}

```

(C)

```

public Pidget(int w1, double w2, int p1, double p2)
{
    pidgetCount = p1;
    pidgetCost = p2;
    super(w1,w2);
}

```

(D)

```

public Pidget(int w1, double w2, int p1, double p2)
{
    widgetCount = w1;
    widgetCost = w2;
    super(p1,p2);
}

```

32. Consider the program segment and class declarations.

```

Widget widget = new Pidget(100,200,300);

```

```

public Kidget
{
    private int kidgetCount;
    public Kidget(int kC)
    {
        kidgetCount = kC;
    }
}

```

```

public Widget
{

```

```

private int widgetCount;
public Widget(int kC, int wC)
{
    super(kC);
    widgetCount = wC;
}
}

public class Pidget extends Widget
{
    private int pidgetCount;
}

```

Which of the following **Pidget** constructors correctly initializes the instances variables?

(A)

```

public Pidget(int kC, int wC, int pC)
{
    super(kC,wC);
    pidgetCount = pC;
}

```

(B)

```

public Pidget(int kC, int wC, int pC)
{
    pidgetCount = pC;
    super(kC,wC);
}

```

(C)

```

public Pidget(int kC, int wC, int pC)
{
    kidgetCount = kC;
    widgetCount = wC
    pidgetCount = pC;
}

```

(D)

```

public Pidget(int kC, int wC, int pC)
{
    super(pC);
    kidgetCount = kC;
    widgetCount = wC
}

```

### *Objective 5 - super Calling a Superclass Method*

33. What happens to a superclass method when it is re-defined in a subclass?

- (A) The superclass method is no longer available.
- (B) The superclass method must be removed to avoid a compile error.
- (C) Both methods in the superclass and subclass are available.
- (D) The superclass method is only available with a superclass object.

34. Method **boo** is defined in super class **Alpha** and **boo** is re-defined in subclass **Beta**. Consider the following program segment.

```
Beta beta = new Beta();  
beta.boo();
```

Which method(s) get called as a result of executing the code segment?

- (A) **boo** defined in **Alpha**, followed by **boo** defined in **Beta**
- (B) **boo** defined in **Beta**, followed by **boo** defined in **Alpha**
- (C) **boo** defined in **Alpha** only
- (D) **boo** defined in **Beta** only

35. Consider the following method, which is defined in the **Student** class and the **Person** class. Assume that the **Student** class is a subclass of the **Person** class.

```
public void showData()  
{  
    System.out.println(getData());  
    System.out.println( super.getData());  
}
```

What is printed when method **showData** is called?

- (A) Two identical values
- (B) A compile error message
- (C) The value of the subclass **getData** followed by the value of the superclass **getData**
- (D) The value of the superclass **getData** followed by the value of the subclass **getData**

36. Consider the following code segment, class **Xerson**, class **Person** and class **Student**.

```
Student tom = new Student(12,15,17);  
tom.showData();  
System.out.println();
```

```
class Xerson  
{  
    private int xer;  
  
    public Xerson(int a)  
    {  
        xer = a;
```

```
class Student extends Person  
{  
    private int grade;  
  
    public Student(int a, int b, int c)  
    {  
        super(a,b);  
        grade = c;  
    }  
  
    public int getData()
```



42. What is the output of this program?

```
public class Java0909
{
    public static void main(String args[])
    {
        Student tom = new Student();
        tom.showData();
    }
}

class Person
{
    protected int age;
    public Person() { age = 18; }
    public getData() { return age; }
}

class Student extends Person
{
    private int grade;
    public Student() { grade = 12; }
    public getData() { return grade; }
    public void showData()
    {
        System.out.println("Grade "+getData());
        System.out.println("Age   "+getData());
    }
}
```

- (A) Grade 12  
Age 18
- (B) Grade 18  
Age 12
- (C) Grade 12  
Age 12
- (D) Age 18  
Grade 18
- (E) Error

43. What is the output of this program?

```
public class Java0910
{
    public static void main(String args[])
    {
        Student tom = new Student();
        tom.showData();
    }
}

class Person
{
    protected int age;
    public Person() { age = 18; }
    public getData() { return age; }
}

class Student extends Person
{
    private int grade;
    public Student() { grade = 12; }
    public getData() { return grade; }
    public void showData()
    {
        System.out.println("Grade "+getData());
        System.out.println("Age   "+super.getData());
    }
}
```

- (A) Grade 12  
Age 18
- (B) Grade 18  
Age 12
- (C) Grade 12  
Age 12
- (D) Age 18  
Grade 18
- (E) Error



44. Look at the program below.  
What commands should be used in place of the *missing commands* to allow the program to work properly?

```
public class Java0911
{
    public static void main(String args[])
    {
        Student tom = new Student(12,18);
        tom.showData();
    }
}

class Person
{
    private int age;
    public Person(int a) { age = a; }
    public getAge() { return age; }
}

class Student extends Person
{
    private int grade;
    public Student(int a, int g) { missing commands }
    public getGrade() { return grade; }
    public void showData()
    {
        System.out.println("Grade "+getGrade());
        System.out.println("Age  "+getAge());
    }
}
```

**Desired Output**  
Grade 12  
Age 18

- (A) super(a);  
grade = g;
- (B) super(g);  
age = a;
- (C) grade = g;  
super(a);
- (D) age = a;  
super(g);

45. Look at the program below.  
What commands should be used in place of the *missing commands* to allow the program to work properly?

```
public class Java0912
{
    public static void main(String args[])
    {
        Car car = new Car("Ford",350);
    }
}

class Engine
{
    private int horsepower;
    public Engine(int hp) { horsepower = hp; }
}

class Car
{
    String type;
    Engine engine;
    public Car(String t, int hp)
    {
        missing
        commands
    }
}
```

- (A) type = t;  
horsePower = hp;
- (B) type = t;  
super(hp);
- (C) super(hp);  
super(t);
- (D) super(t);  
horsePower = hp;
- (E) type = t;  
engine = new Engine(hp);

---

46. Assume these 2 classes are in the same program.

```
class Tomato
{
}

class Microwave extends Tomato
{
}
```

Which of these statements does NOT construct an object properly?

- (A) Microwave bob = new Microwave();
- (B) Tomato bob = new Tomato();
- (C) Microwave bob = new Tomato();
- (D) Tomato bob = new Microwave();

---

47. What is the name of the class that ALL classes inherit from automatically?

- (A) Object
- (B) Class
- (C) extends
- (D) Inheritance
- (E) Composition

---

48. When a subclass has a method with the same signature as the superclass, what is that called?

- (A) instantiation
  - (B) composition
  - (C) overriding
  - (D) unnecessary
-

---

49. What is the keyword **super** used for in Java?

- I. It calls superclass constructor.
- II. It allows you to call a superclass method when the subclass has a method with the same identifier.
- III. It allows you to format your output to display "superscript" for things like exponents.

- (A) I only
  - (B) II only
  - (C) III only
  - (D) I and II only
  - (E) I, II and III
- 
-