

## OOPS

1. 

```
public class SportsReference {
    public static void main(String[] args) {
        Sport ref;
        Baseball aBaseball = new Baseball();
        Basketball aBasketball = new Basketball();
        Football aFootball = new Football();
        ref = aBaseball; ref.color();
        ref = aBasketball; ref.color();
        ref = aFootball; ref.color();
    }
}
```
- Which type of polymorphism is used in the sample code above?
- a. Dynamic Method Binding
  - b. Parametrics
  - c. Overloaded Methods
  - d. Overridden Methods
  - e. Variable Methods

2. Sample Code
- ```
// Assume DateValue is a class that represents a date
// in some unspecified format.
public void printIfEqual(DateValue dv1, DateValue dv2) {
    if (dv1==dv2) {
        System.out.println("Dates "+dv1+" and "+dv2
            +"are the same day");
    }
}
```
- What is the potential usage error in the sample code above?
- a. The dv1==dv2 comparison fails if DateValue does not override the equals() method.
  - b. System.out.println() throws an IOException that is not caught or rethrown by printIfEqual.
  - c. The two objects represent the same date but do not satisfy dv1==dv2.
  - d. The println statement fails if DateValue does not implement toString().
  - e. System.out is redirected away from the console.

3. You override the finalize() method when:
- a. an object needs to be written on disk.
  - b. there is a cleanup activity needed before an object is created.
  - c. You want to kill a thread.
  - d. You instantiate a final class.
  - e. The object is declared on a heap, so its memory may be reclaimed.

## OOPS

4. Line 1 class sample {  
Line 2     void example() {  
Line 3         System.out.println("sample case");  
Line 4     }  
Line 5 }  
Line 6 public class model extends sample {  
Line 7     void example() {  
Line 8         System.out.println("model case");  
Line 9     }  
Line 10    private static void main(string args[]) {  
Line 11        model d = new model();  
Line 12        sample a = new sample();  
Line 13        d.example();  
Line 14        a.example();  
Line 15    }  
Line 16 }

Given the above sample code, what needs to be changed for the code to execute?

- a. Change Line 2 to:  
    static example() {
- b. Change Line 6 to :  
    Private class model extends sample {
- c. Change Line 10 to:  
    public class void main(String[] args){
- d.** Change Line 10 to:  
    public static void main(String[] args){
- e. Change Line 10 to:  
    public static void case(String[] args){

5. private String thing;  
String getThing(){  
    return thing;  
}

Based on the above sample code, what do you add to the declaration of getThing() method for the method to be callable within a non-subclass class in a different package?

- a. private
- b. protected
- c. super
- d.** public
- e. transient

## OOPS

```
6. Line 1 public class Test {  
    Line 2     public static void main(string[] args) {  
    Line 3         int x = getValue();  
    Line 4         System.out.println(x);  
    Line 5     }  
    Line 6     public static getValue() {  
    Line 7         return 10;  
    Line 8     }  
    Line 9 }
```

What change do you make to the sample code above to for it execute?

- a. Change Line 2 to  
private static void main(string[] args
- b. Change Line 2 to  
public static void(main)
- c. Change Line 3 to  
int x = main(String[] args)
- d. Change Line 6 to  
private static getValue() {
- e.** Change Line 6 to  
private static int getValue() {

```
7. public class Main{  
    private void example()  
    {  
        System.out.println("sample example");  
    }  
    public static void main(String[] args) {  
        Main obj = new Main();  
        obj.example();  
    }  
}
```

Assuming the main method is static, why does the sample code above not produce a compile error?

- a. Member methods of a class can access public members of the same class regardless of their modifiers.
- b. The method is called from outside the class, and a class can use its public members.
- c.** The method is invoked using the instance class obj.
- d. The access modifier allows the private method to execute as long as the outside class is set to public.
- e. The abstract methods are implemented in an abstract class.

## 8. Sample Code

```

class Animal{
    void speak(){ System.out.println("speak"); } }
class Dog extends Animal{
    void speak(){System.out.println("woof!"); } }
class Cat extends Animal{
    void speak(){System.out.println("meow!"); } }
public class AnimalTest{
    public static void main( String[] args ){
        Animal[] animals = new Animal[3];
        animals[0] = new Animal();
        animals[1] = new Cat();
        animals[2] = new Dog();
        for( int i=0; i < animals.length; i++ )
            animals[i].speak();
    }
}

```

What is the result of executing the sample code above?

- a. speak  
meow!  
woof!
- b. woof!  
speak  
meow!  
Speak
- c. speak  
meow!  
speak  
woof!
- d. speak  
speak  
speak
- e. speak  
woof!  
meow!  
Speak

**9. Sample Code**

```
import java.io.*;
public class TestScope{
    public static void main(String args[]) {
        int i=0,k=0;
        for (i=1;i<20;i++) {
            int j=i*2;
            k=j*k;
            System.out.println(j);
        }
        for (int m=1;m<10;m++) {
            System.out.println(m);
        }
        System.out.println("Value is "+m);
    }
}
```

Which line in the sample code above contains the compile-time error?

- a. for (i=1;i<20;i++) {
- b. int j=(i\*2);
- c. for (int m=1;m<10;m++) {
- d. System.out.println(m)
- e. System.out.println("Value is " + m);**

**10. Sample Code**

```
public boolean testVal(int a) {
    // ???
}
```

Which line of code do you insert in place of ??? in the sample code above for the function to return true, if the value of "a" is greater than 5 and false otherwise?

- a. do {if (a>5) true else false};
- b. return (a<5)?true:false;
- c. do { return false } if (a<=5);
- d. return (a>5);**
- e. if (a<=5) {return=false;} else {return=false;};

```
11. class A {  
    int i=0;  
    public A() { i=8; }  
    public static void main(String args[]) {  
        int i = 0;  
        A h = new A();  
        while (h.i <= 10)  
            h.doIt();  
    }  
    public static void doIt() {  
        i++;  
        System.out.println("Hello");  
    }  
}
```

What is the result of the sample code above when executed?

- a. It prints "Hello" 2 times.
- b. It prints "Hello" 3 times.
- c. It prints "Hello" 11 times.
- d. It does not compile because variable i has been declared twice.
- e. It does not compile because doIt() cannot reference non-static variable i.**

12. What do you define in an interface?

- a. Transient variables
- b. Instance variables
- c. Final methods
- d. Constants**
- e. Static methods

## OOPS

```
13. public class A {  
    public void a(){  
        System.out.println("class A");  
    }  
}  
public class B extends A{  
    public void a(){  
        System.out.println("class B");  
    }  
}  
public class C extends B{  
    public void a(){  
        super.a();  
        System.out.println("class C");  
    }  
}  
public class D {  
    public static void main(String[] args) {  
        A c=new C();  
        c.a();  
    }  
}
```

What is the output of the sample code above?

- a. Class A  
Class C
- b. Class C  
Class B
- c. class A  
class B
- d. class B  
class C**
- e. Class B  
Class A

**14. Sample Code**

```
Line 1 public class Sample {  
Line 2     private String un;  
Line 3     private String te;  
Line 4     public Sample(String unit, String test){  
Line 5         un = unit;  
Line 6         te = test;  
Line 7         myExample = new Example(100);  
Line 8     }  
Line 9 }
```

Assuming the proper imports, what do you change to allow the sample code above to compile and be used by another class to create an instance of the class Sample?

- a. Insert `public void main(String[] args) {}` after Line 8.
- b. Change Line 2 to:  
`public un;`
- c.** Insert `private Example myExample;` after Line 2.
- d. Change Line 7 to:  
`myExample = new example.count < 100)`
- e. Remove `new Example(100);` from Line 7.



**15. Sample Code**

```
class SuperClass {  
    public void printIt() {  
        System.out.println("SuperClass");  
    }  
    public void printIt(boolean print) {  
        if (print) {  
            System.out.println("Super-part 2");  
        }  
        else {  
            printIt();  
        }  
    }  
}  
class SubClass extends SuperClass {  
    public void printIt() {  
        System.out.println("SubClass");  
    }  
}  
public class TestSub {  
    public static void main(String args[]) {  
        SuperClass sc=new SubClass();  
        sc.printIt();  
        sc.printIt(false);  
    }  
}
```

What is the output of the sample code above?

- a. SubClass  
    Super-part 2
- b. SuperClass  
    SubClass
- c. SubClass  
    SuperClass
- d. SubClass  
    SubClass**
- e. SuperClass  
    SuperClass

## OOPS

```
16. public class TestOrder {  
    static int deck[]=new int[25];  
    static {  
        for (int j=0;j<deck.length;j++)  
            deck[j]=j;  
        System.out.println("Finished creating the deck");  
    }  
    public static void main(String args[]) {  
        System.out.println("Creating an Object");  
        TestOrder to=new TestOrder();  
        System.out.println("Finished creating");  
    }  
}
```

What is the output of the sample code above?

- a. Creating an Object  
Finished creating  
Finished creating the deck
- b. Finished creating the deck  
Creating an Object  
Finished creating**
- c. Finished creating  
Creating an Object  
Finished creating the deck
- d. Creating an Object  
Finished creating the deck  
Creating an Object
- e. Finished creating the deck  
Finished creating  
Creating an Object

## OOPS

```
17. class Brain {
    Brain(int a, int b) {
        int c = a + b;
        System.out.println("Sum=" + c);
    }
    void display() {
        System.out.println("Statement");
    }
}
class Bench extends Brain {
    Bench(int a, int b) {
        int c = a - b;
        System.out.println("Difference=" + c);
    }
}
public class Thread {
    public static void main(String args[]) {
        Bench c = new Bench(2, 1);
        c.display();
    }
}
```

Why does the sample code above fail to compile?

- a. Thread should be declared prior to Bench for Bench to be public.
- b. Brain is not adequately called as a method or class.
- c. A constructor must be explicitly invoked on Bench.**
- d. Bench is the parent of Brain and must be declared first.
- e. All classes are private unless specifically declared otherwise.

**18. Sample Code**

```
public class One {
    public String getX() {
        return "One";
    }
    public static void main(String args[]) {
        One one = new Five();
        System.out.println(one.getX());
    }
}
class Two extends One {
    public String getX() {
        return "Two";
    }
}
class Three extends Two {
    public String getX() {
        return "Three";
    }
}
class Four extends Three {
    public String getX() {
        return "Four";
    }
}
class Five extends Four {
    public String getX() {
        return "Five";
    }
}
```

What is printed when you execute the sample code above?

- a. One
- b. Two
- c. Three
- d. Four
- e. Five**

## OOPS

19. Line 1 public class MyString {  
Line 2     public static void main(String args[]) {  
Line 3         System.out.println("Hello, World");  
Line 4     }  
Line 5 }

Which line of code do you use to replace Line 2 in the sample code above in order to print "Hello, World" when you pass just the class name to the Java command?

- a. public static main void(String []args{}) {
- b. public static main void(String[] args) {
- c. public static void main(String args...) {
- d. public static void main(String... args) {**
- e. static public void main(String args[\*]) {

20. Which one of the following is a valid declaration in an interface?

- a. public abstract static int getPsn();
- b. protected int getPsn();
- c. static final int psno = 3;**
- d. public static int getPsn();
- e. public static transient int psno = 3;

21. Which lists access modifiers ranked from LEAST visibility to MOST?

- a. public, <none>, protected, private
- b. public, protected, <none>, private
- c. <none>, private, protected, public
- d. private, <none>, protected, public**
- e. private, protected, <none>, public

22. Scrollable is an interface.

-Writable is an interface.

-TextScreen is an abstract class

VideoScreen is a class that extends TextScreen and implements both Scrollable and Writable.

In reference to the scenario above, which one of the following statements is always true?

- a. A Scrollable object can be cast to produce a VideoScreen.
- b. A VideoScreen object cannot be passed to a method that expects a Scrollable
- c. A VideoScreen object cannot be passed to a method that expects a Writable
- d. A VideoScreen object cannot be passed to a method that expects a TextScreen
- e. A VideoScreen object can be cast as Scrollable**

## OOPS

23. What is the result of the following code?

```
1. public class Shape {  
2.     private String color;  
3.  
4.     public Shape(String color) {  
5.         System.out.print("Shape");  
6.         this.color = color;  
7.     }  
8.  
9.     public static void main(String [] args) {  
10.        new Rectangle();  
11.    }  
12. }  
13.  
14. class Rectangle extends Shape {  
15.     public Rectangle() {  
16.         System.out.print("Rectangle");  
17.     }  
18. }
```

- a. ShapeRectangle
- b. RectangleShape
- c. Rectangle
- d. Line 4 generates a compiler error.
- e. Line 15 generates a compiler error.**

## OOPS

**24.** Given the following class definitions:

```
1. public class Parent {  
2.     public Parent() {  
3.         System.out.print("A");  
4.     }  
5. }  
6.  
7. class Child extends Parent {  
8.     public Child(int x) {  
9.         System.out.print("B");  
10.    }  
11.  
12. public Child() {  
13.     this(123);  
14.     System.out.print("C");  
15. }  
16.}
```

what is the output of the following statement?  
new Child();

- a.** ABC
- b. ACB
- c. AB
- d. AC
- e. This code does not compile.

**25.** What is the output of the following program?

```
1. public class WaterBottle {  
2.     private String brand;  
3.     private boolean empty;  
4.  
5.     public static void main(String [] args ) {  
6.         WaterBottle wb = new WaterBottle();  
7.         if(!wb.empty) {  
8.             System.out.println("Brand = " + wb.brand);  
9.         }  
10.    }  
11.}
```

- a. Line 6 generates a compiler error.
- b. Line 7 generates a compiler error.
- c. Line 8 generates a compiler error.
- d. There is no output.
- e.** Brand = null

## OOPS

26. Given the following class definition:

```
1. public class Television {  
2.     private int channel = setChannel(7);  
3.  
4.     public Television(int channel) {  
5.         this.channel = channel;  
6.         System.out.print(channel + " ");  
7.     }  
8.  
9.     public int setChannel(int channel) {  
10.        this.channel = channel;  
11.        System.out.print(channel + " ");  
12.        return channel;  
13.    }  
14.}
```

what is the output of the following statement?  
new Television(12);

- a. 12
- b. 12 7
- c. 7 12**
- d. 7
- e. The code does not compile.



## OOPS

27. Given the following my.school.ClassRoom and my.city.School class definitions:

```
1. //ClassRoom.java
2. package my.school;
3. public class ClassRoom {
4.     private int roomNumber;
5.     protected String teacherName;
6.     static int globalKey = 54321;
7.
8.     ClassRoom(int r, String t) {
9.         roomNumber = r;
10.        teacherName = t;
11.    }
12. }
//School.java
1. package my.city;
2. import my.school.ClassRoom;
3. public class School {
4.     public static void main(String [] args) {
5.         System.out.println(ClassRoom.globalKey);
6.         ClassRoom room = new ClassRoom(101, "Mrs.Anderson");
7.         System.out.println(room.roomNumber);
8.         System.out.println(room.teacherName);
9.     }
10. }
```

which of the following line numbers in main generate a compiler error? (Select all that apply.)

- a. None; the code compiles fine.
- ☒ b. Line 5
- ☒ c. Line 6
- ☒ d. Line 7
- e. Line 8

## OOPS

**28.** Given the following interface and class defined in a file named `Traceable.java`, what is the result of compiling this code?

```
1. public interface Traceable {  
2.     public static int MAX_TRACE;  
3.     public void trace();  
4. }  
5.  
6. class Picture implements Traceable {  
7.     public void trace() {  
8.         System.out.println("Tracing a picture");  
9.     }  
10. }
```

- a. Two bytecode files: `Traceable.class` and `Picture.class`
- b. One bytecode file: `Traceable.class`
- c. Compiler error on line 2**
- d. Compiler error on line 3
- e. Compiler error on line 6
- f. Compiler error on line 7

**29.** Given the following class definitions:

```
1. class Parent {  
2.     public void printResults(String... results) {  
3.         System.out.println("In Parent");  
4.     }  
5. }  
6.  
7. class Child extends Parent {  
8.     public int printResults(int id) {  
9.         System.out.println("In Child");  
10.        return 0;  
11.    }  
12.}
```

what is the result of the following statement?  
`new Child().printResults(0);`

- a. In Parent
- b. In Child**
- c. 0
- d. Line 2 generates a compiler error.
- e. Line 8 generates a compiler error.

## OOPS

30. What is the result of the following program?

```
1. class Parent {
2.     public float computePay(double d) {
3.         System.out.println("In Parent");
4.         return 0.0F;
5.     }
6. }
7.
8. public class Child extends Parent {
9.     public double computePay(double d) {
10.        System.out.println("In Child");
11.        return 0.0;
12.    }
13.
14. public static void main(String [] args) {
15.     new Child().computePay(0.0);
16. }
17. }
```

- a. In Parent
- b. In Child
- c. 0.0
- d. null
- e. The code does not compile.**

31. Suppose a method in a class has the following method declaration:

```
public java.io.OutputStream createStream(String fileName) {
    //method body here...
}
```

Which of the following methods could appear in a child class and override createStream ?

- a. public java.io.OutputStream createStream(String f)**
- b. public java.io.OutputStream createStream(char c)
- c. public java.io.FileOutputStream createStream(String f)**
- d. public void createStream(String c)
- e. public java.io.OutputStream createStream(StringBuffer fileName)
- f. protected java.io.OutputStream createStream(String fileName)

## OOPS

**32.** Given the following class definitions,  
what is the output of the statement `new Child();` ?

```
1. class Parent {
2. {
3. System.out.print("1");
4. }
5.
6. public Parent(String greeting) {
7. System.out.print("2");
8. }
9. }
10.
11. class Child extends Parent {
12. static {
13. System.out.print("3");
14. }
15.
16. {
17. System.out.print("4");
18. }
19. }
```

- a. 1234
- b. 3123
- c. 3142
- d. 3124
- e.** The code does not compile

**33.** Given the following interface definitions:

```
1. //Readable.java
2. public interface Readable {
3.     public abstract void read();
4. }

1. //SpellCheck.java
2. public interface SpellCheck extends Readable {
3.     public void checkSpelling();
4. }
```

which of the following statements are true? (Select all that apply.)

- a. The SpellCheck interface does not compile.
- b. A class that implements Readable must override the read method.
- c.** A class that implements SpellCheck inherits both the checkSpelling and read methods.
- d. A class that implements SpellCheck only inherits the checkSpelling method.
- e. An interface cannot extend another interface.

## OOPS

34. Which one of the following method declaration is valid?

- a. static public void ( ) { }
- b. static public protected intgetInt() { return 2; }
- c. static public intgetInt() { return Integer.SIZE; }**
- d. public static intgetInt() { return Math.PI; }
- e. public static int() { return 2; }

35. Given the below code

```
Class Base{}
class Derived1 extends Base{}
class Derived2 extends Base{}
class Test{
    public static void main(String[]args){
        Base b = new Base();
        Derived1 d1 = new Derived1();
        Derived2 d2 = newDerived2();
        //assignment here
    }
}
```

Which of the following assignments is legal?

- a. d1 = b
- b. d2 = d1
- c. d2 = b
- d. b = d1**
- e. d1 = d2