//

IJSE

Institute of Software Engin

Graduate Diploma in Software Engineering

Batch - GDSE69

Module - Object Oriented Programming

Name : Shimara Appuhami

Nic : 200362310434

//
/////////

Assignment 01

01. Briefly explain of followings terms with appropriate examples.

a. **Class/ Template**

- A class or template is a blueprint for creating objects in object-oriented programming. It defines a set of attributes and methods that will be shared by all instances (objects) created from it.

```
● public class student {
● String id;
● String name;
●
● void write() {
● System.out.println("Student is writing");
● }
● }
```

b. **Object/ Instance**

- A specific entity with its unique set of attributes and can perform actions defined by the methods of its class.

```
● student s1 = new Car();
● s1.id = "s001";
● s1.name = "kaml";
```

c. **Methods/ Functions**

- Methods or functions are blocks of code within a class or elsewhere that perform specific tasks. Methods define the behaviour of an object.

```
● public class student {
● int add(int id, int name) {
● return “id”+id+”name”+name;
● }
● }
```

d. **Attributes/ Properties**

- Attributes or properties are the characteristics or data associated with an object. They represent the state of an object.

```
● public class student {
● String id;
● int age;
● }
```

e **Reference Variables**

- Reference variables hold references (memory addresses) to objects. They point to the location in memory where the object is stored.

- student s1 = new student();

f. **Primitive Variables**

- Primitive variables hold simple, single values and are not objects. They are basic data types like integers, floating-point numbers, and characters.

- int num = 5;

g. **Method Parameters**

- Method parameters are variables that are passed into a method when it is called. They allow the method to operate on specific data.

- public class total {
- int add(int a, int b) {
- return a + b;
- }
- }

h. **Local Variables**

- Local variables are variables declared within a specific scope, such as within a method. They exist only within that scope.

- public void exampleMethod() {
- int id = 10;
- }

i. **Default Values**

- Default values are the initial values assigned to variables if no explicit value is provided. They ensure that variables have a valid starting point.

- public class Example {
- int number; // 'number' will have a default value of 0
- }

j. **Declaration Values.**

- Declaration values are the initial values assigned to variables at the time of declaration.

- int x = 10;

//

02. Given Code:

```
//--------------------Test.java-----------------------
class Test{
int a=8;
int b=9;
}
//--------------------Demo.java-----------------------
class Demo{
public static void main(String args[]){
System.out.print(a);
System.out.println(b);
}
}
```

What is the result of attempting to compile & run the program?

A. 89

B. 00

c. 88

D. Compile time error

E. None of the above

//

03. Given Code:

```
//--------------------Test.java-----------------------
class Test{
int a=8;
int b=9;
}
//--------------------Demo.java-----------------------
class Demo{
public static void main(String args[]){
Test t1=new Test();
System.out.print(t1.a);
System.out.println(t1.b);
}
}
```

What is the result of attempting to compile & run the program?

A. 89

B. 00

C . 88

D. Compile time error

//

04. Briefly explain the difference between “attributes” and “methods” in java Object using real world examples.

- Attributes, also known as fields or properties, represent the characteristics or data associated with an object. They define the state of the object.
- Eg-: "Person" example,
  attributes like name,
  age, and address
  describe the person

- Methods are functions or procedures associated with an object. They define the behavior or actions that an object can perform.
- Eg-: methods like talk() and walk()
  represent actions that the
  person can take.

//

05. Which of the following lines are illegal? Explain your answer.

```
class Box{
int length;
int width;
int height;
}
class Demo{
public static void main(String args[]){
Box b1=new Box(); // 1
System.out.println("length of box : "+length); //2
System.out.println("width of box : "+width); //3
System.out.println("height of box : "+height); //4
}
}
```

- Line 1>>>>legal
- Line 2>>>>illegal
- Line 3>>>>illegal
- Line 4>>>>illegal
- (added b1. before each occurrence of length, width, and height to indicate that these variables are accessed through the instance b1 of the Box class.)

//

06. What is the output of the following program? Explain your answer.

```
//--------------------Box.java-----------------------
class Box{
int length;
int width;
int height;
void printVolume(){
int volume;
volume=length*width*height;
System.out.println("Volume : "+volume);
}
}
//--------------------Demo.java-----------------------
class Demo{
public static void main(String args[]){
Box b1=new Box();
b1.printVolume();
System.out.println("length of box : "+b1.length);
System.out.println("width of box : "+b1.width);
System.out.println("height of box : "+b1.height);
}
}
```

- Volume : 0
- length of box : 0
- width of box : 0
- height of box : 0
- If no values are given,the compiler will use default values

//

07. What is the output of the following program? Explain your answer.

```
//--------------------Box.java-----------------------
class Box{
int length=12;
int width=5;
int height=3;
void printVolume(){
int volume;
volume=length*width*height;
System.out.println("Volume : "+volume);
}
}
//--------------------Demo.java-----------------------
```

```
class Demo{
public static void main(String args[]){
Box b1=new Box();
b1.printVolume();
System.out.println("length of box : "+b1.length);
System.out.println("width of box : "+b1.width);
System.out.println("height of box : "+b1.height);
}
}
```

- Volume : 180
- length of box : 12
- width of box : 5
- height of box : 3
- The printVolume()method calculates and print 12 x 5 x 3=180;

//

08. What is the output of the following program? Explain your answer.

```
//--------------------Box.java-----------------------
class Box{
int length;
int width;
int height;
Box(){
System.out.println("default constructor");
length=2;
width=2;
height=2;
}
}
//--------------------Demo.java----------------------
class Demo{
public static void main(String args[]){
Box b1=new Box();
System.out.println("length of box : "+b1.length);
System.out.println("width of box : "+b1.width);
System.out.println("height of box : "+b1.height);
}
}
```

- default constructor
- length of box : 2
- width of box : 2

- height of box : 2
- Object is created in new Box() and prints "default constructor"and sets the values length,height,width to 2.

//

09. What is the output of the following program? Explain your answer.

```
//--------------------Box.java-----------------------
class Box{
int length;
int width;
int height;
Box(int l,int w,int h){
System.out.println("Parameterized constructor");
length=l;
width=w;
height=h;
}
void printVolume(){
int volume;
volume=length*width*height;
System.out.println("Volume : "+volume);
}
}
//--------------------Demo.java-----------------------
class Demo{
public static void main(String args[]){
Box b1=new Box(5,4,3);
b1.printVolume();
Box b2=new Box(12,5,3);
b2.printVolume();
}
}
```

- Parameterized constructor
- Volume : 60
- Parameterized constructor
- Volume : 180

//

10. What is the output of the following program? Explain your answer.

```
class Box{
int length;
```

```
private int width;
int height;
}
//--------------------Demo.java-----------------------
class Demo{
public static void main(String args[]){
Box b1=new Box();
System.out.println("length of box : "+b1.length);
System.out.println("width of box : "+b1.width);
System.out.println("height of box : "+b1.height);
}
}
```

- The width variable in the Box class is declared as private, which means it is not directly accessible outside the Box class. Therefore, attempting to access it in the Demo class will result in a compilation error.
- length of box :0;
- height of box :0;

//

11. Which of the following lines are illegal? Explain your answer.

```
class Box{
int length; //Line 1
int width; //Line 2
int height; //Line 3
length=12; //Line 4
width=5; //Line 5
height=3; //Line 6
}
```

- Line 1 >>>>legal
- Line 2 >>>>legal
- Line 3 >>>>legal
- Line 4 >>>>illegal
- Line 5 >>>>illegal
- Line 6 >>>>illegal
- Java, statements that are not part of a method or constructor body are not allowed.

//

12. What is the purpose of the constructor in a java Template (Class)? Explain your answer the suitable Examples.

- Constructor in java is used to create the instance of the class. Constructors are almost

similar to methods except for two things its name is the same as the class name and it has no return type.

```
class Student {
double prf;
double dbm;

public Student() {
System.out.println("Constructor calling!");
}
}

public class Main {
public static void main(String args[]) {
Student s1 = new Student();

new Student();

Student s2;
}
}
```

//

13. Briefly explain the difference between “constructor s” and “methods” in java Object using real world Examples.

- Constructors do not return any type while method have the return type or void if it does not return any value. Constructors are called only once at the time of Object creation while method can be called any number of times

```
public class Car {
String make;
String model;
int year;

// Constructor
public Car(String make, String model, int year) {
this.make = make;
this.model = model;
this.year = year;
System.out.println("A new car has been created!");
}
}
```

-
-
- Car myCar = new Car("Toyota", "Camry", 2022);

//

14. Which of the following lines of code could be inserted at line 12 and still allow the code to compile?

```
class Box{
int length;
int width;
int height;
}
class Demo{
public static void main(String args[]){
//Insert code here
}
}
```

A. Box b1;
B. Box b1=new Box();
C. new Box();
D. Box b1=new Box(12,5,3);
E. Box b1=new Box(10);

//

15. What is the output of following program? Explain your

```
class Box{
int length;
int width;
}
class Demo{
public static void main(String args[]){
Box b1=new Box();
Sustem.out.print(b1.length); //Line 1
Sustem.out.print(b1.WIDTH); //Line 2
Sustem.out.print(b1.height); //Line 3
}
}
```

- Line 1>>>>0
- Line 2>>>>error
- Line 3>>>>error

//

16. What is the output of the following program?

```
class Test{
int a=3;
int b=4;
Test(int i,int j){
a=i;b=j;
}
Test(){}
}
class Demo{
public static void main(String args[]){
Test t1=new Test(1,2);
System.out.print(t1.a+" "+t1.b+" ");
Test t2=new Test();
System.out.println(t2.a+" "+t2.b);
}
}
```

A. 4 4 2 2
B. 4 3 2 1
C. 1 2 3 4
D. 4 3 2 1

//

17. Which of the following lines of code could be inserted at line 10 to get the following output?
Prints “Values 100 200”

```
class MyClass{
/* * Insert Code Here Line 10 */
void printValues(){
System.out.println("Values : "+x+", "+y);
}
}
class Demo{
public static void main (String []args){
MyClass c=new MyClass(100,200);
c.printValues();
}
}
```

A. int x;
int y;
MyClass(int i, int j){ x=i; y=j; }
//=======================
B. int x=100;
int y=200;

```
MyClass(int i, int j){ }
//=========================
C. int x;
int y;
MyClass(int i,int j){ i=x; j=y; }
//=========================
D. int x;
int y;
MyClass(int i, int j){ x=100; y=200; }
```

//

18. What will be the output when you compile and run the following program? Explain your answer.

```
class A{
int a;
A(int i){ a=i; }
}
class Demo{
public static void main(String args[]){
A a1=new A(100);
A a2=new A(101);
System.out.println(a1.a+" "+a2.a);
a1=a2;
a1.a=0;
System.out.println(a1.a+" "+a2.a);
}
}
```

- 100 101
- 0 0

//

19. What will be the output when you compile and run the following program? Explain your answer.

```
class Item{
int code;
Item(int i){ code=i;}
void printCode(int code){
System.out.println("Code : "+code);
}
}
class Demo{
public static void main(String args[]){
Item t1=new Item(2001);
t1.printCode(3001);
```

```
}
}
```

- Code : 3001

//

20. What will be the output when you compile and run the following program? Explain your answer.

```
class Item{
int code;
Item(int i){ code=i;}
void printCode(int code){
System.out.println("Code : "+this.code);
}
}
class Demo{
public static void main(String args[]){
Item t1=new Item(2001);
t1.printCode(3001);
}
}
```

- Code : 3001

//

21. Which of the following lines of code could be inserted at line 12 and still allow the code to compile? Explain your answers.

```
class A{
int a;
//Line 12
}
class Demo{
public static void main(String args[]){
A a2=new A();
}
}
```

A. A(int i){a=i;}
B. void A(int i){a=i;} // **compile**
C. A(){} //**compile**
D. Insert nothing

//

22. Which of the following lines of code could be inserted at line 12 to get output “100” for the following java

application? Explain your answers.

```
class A{
int a;
//
void printValue(){
System.out.println(a);
}
}
class Demo{
public static void main(String args[]){
A a1=new A(100);
a1.printValue();
}
}
```

A. A(int i){a=i;}
B. A(int a){a=a;}
C. A(int a){a=this.a;}
D. A(int a){this.a=a;}
E. A(int i){a=100;} //100 but not passed value
F. A(int i){i=a;}

//

23. Describe “encapsulation” the key concept in the Object Oriented Programming with appropriate examples.

- Encapsulation is one of the fundamentals of OOP (object-oriented programming). It refers to the bundling of data with the methods that operate on that data. Encapsulation is used to hide the values or state of a structured data object inside a class, preventing unauthorized parties' direct access to them.

- Data Hiding
- Access Control
- Security
- Modularity

- class Item {
- private String code;
- private String description;
- private double unitPrice;
- private int qtyOnHand;
-
- public Item() {
- }

●
● public Item(String code, String description, double unitPrice, int qtyOnHand) { ●
//values injection
● this.code = code;
● this.description = description;
● this.unitPrice = unitPrice;
● this.qtyOnHand = qtyOnHand;
● }
●
● public double getDiscount(double disc) {
● return unitPrice - disc;
● }
●
●
● //getters and setter
●
● public void setCode(String code) { ●
this.code = code;
● }
● public String getCode() {
● return code;
● }
●
● public void setDescription(String description) { ●
this.description = description; ● }
● public String getDescription() {
● return description;
● }
●
● public void setUnitPrice(double unitPrice) { ●
this.unitPrice = unitPrice; ● }
● public double getUnitPrice() {
● return unitPrice;
● }
●
● public void setQtyOnHand(int qtyOnHand) { ●
this.qtyOnHand = qtyOnHand; ● }
● public int getQtyOnHand() {
● return qtyOnHand;
● }
●
● }
●

- class ItemCollection {
- Item[] itemList;
- int index;
-
- public ItemCollection(int size) {
- itemList = new Item[size]; ● index = -1;
- }
-
- public void add(Item data) {
- itemList[++index] = data; ● }
- }
-
- public class Main {
- public static void main(String args[]) {
- Item i1 = new Item("I001", "Rice 1KG", 760, 23);
- Item i2 = new Item("I002", "Toffee", 15, 400);
- Item i3 = new Item("I003", "Busicuit", 160, 57);
-
- ItemCollection ic = new ItemCollection(3);
-
- ic.add(i1);
- ic.add(i2);
- ic.add(i3);
- }
- }

//

24. What is difference between “Tightly encapsulated” and “Loosely encapsulated”? Explain your answer with appropriate examples.

- **Tightly Encapsulated:** Attributes are private and access is controlled through getter and setter methods. Internal details are well-protected, providing a high level of security and control.

- class Student { //Tightly Encapsulated class
- private int id;
- private String name;
- private int age;
- }

- **Loosely Encapsulated:** Attributes may have broader access modifiers making them more accessible directly from external code. This reduces the level of control and security over the internal state of the object.

- public class Student {
- public String name; // Loosely encapsulated attribute
- public int age; // Loosely encapsulated attribute
-
- public Student(String name, int age) {
- this.name = name;
- this.age = age;
- }
- }

//

25. Given:
class Test{
int a=100; // Line 1
System.out.print(a); //Line 2
}
class DemoTest{
public static void main(String args[]){
Test t=new Test();
}
}
What is the result of attempting to compile and run the
program?
A. Prints: 100
B. Prints: 0
C. Compiler error at line 1.
D. Compiler error at line 2

//

26. Create fully encapsulated class “Date” with the
following functionalities.
//----------Date.java----------
class Date{

```
int year=1970;
int month=1;
int day=1;
}
//--------------------Demo.java-----------------------
class Demo{
public static void main(String args[]){
Date d1=new Date();
d1.printDate(); //1970-1-1
d1.year=2016; //Illegal
d1.month=5; //Illegal
d1.day=30; //Illegal
/*year, month and day attributes
*cannot be accessed to another class
*/
d1.setYear(2016);
d1.setMonth(5);
d1.setDay(31);
System.out.println("Year : "+d1.getYear());
System.out.println("Month :"+d1.getMonth());
System.out.println("Day : "+d1.getDay()); }
}
```

- class Date{
- private int year=1970;
- private int month=1;
- private int day=1;
- Date(){}
- public Date(int year,int month,int day){
- this.month=month;
- this.month=month;
- this.day=day;
- }
- public void printDate(){
- System.out.print(year+"-"+month+"-"+day); ● }
- public void setYear(int year){
- this.year=year;
- }
- public void setMonth(int month){
- this.month=month;
- }
- public void setDay(int day){
- this.day=day;
- }

-
- public int getYear(){
- return year;
- }
- public int getMonth(){
- return month;
- }
- public int getDay(){
- return day;
- }
- }
- //--------------------Demo.java------------------------
-
- class Main1{
- public static void main(String args[]){
- Date d1=new Date();
- d1.printDate(); //1970-1-1
- //d1.year=2016; //Illegal
- //d1.month=5; //Illegal
- //d1.day=30; //Illegal
- /*year, month and day attributes
- *cannot be accessed to another class
- */
- d1.setYear(2016);
- d1.setMonth(5);
- d1.setDay(31);
- System.out.println("Year : "+d1.getYear());
- System.out.println("Month :"+d1.getMonth());
- System.out.println("Day : "+d1.getDay());
- }
- }

///

27. Explain the following lines of code according to class "Customer" is given bellow.

```
class Customer{
private String id;
private String name;
public Customer(){}
public Customer(String id, String name){
this.id=id;
this.name=name;
}
```

```
public void printCustomer(){
System.out.println(id+" - "+name);
}
public void setCustomerDetail(String id, String
name){
this.id=id;
this.name=name;
}
public void setId(String id){
this.id=id;
}
public void setName(String name){
this.name=name;
}
}
//--------------------DemoCustomer.java--------------------
class DemoCustomer{
public static void main(String args[]){
Customer c1; //Line 1
c1=new Customer("C001", "Danapala");//Line 1
c1.printCustomer(); //Line 3
Customer c2; //Line 4
c2=new Customer(); //Line 5
c2.setCustomerDetail("C002", "Gunapala"); //Lin6
c2.printCustomer(); //Line 7
Customer c3; //Line 8
c3=new Customer(); //Line 9
c3.setId("C003"); //Line 10
c3.setName("Somapala"); //Line 11
c3.printCustomer(); //Line 12
}
}
```

- The customer have 2 private attributes
- The class provides methods to print customer details
- Line 1 >>>> Creates a Customer object c1
- Line 2 >>>> create the parameterized constructor
- Line 3 >>>> sets the customer details, and prints them.
- Line 4 >>>> Creates another Customer object c2
- Line 5 >>>> the default constructor
- Line 6 >>>> sets the customer details using the setCustomerDetail method
- Line 7 >>>> calling printCustomer method.
- Line 8 >>>> Creates another Customer object c3
- Line 9 >>>> create the parameterized constructor,

- Line 10>>> sets the ID and name using the setId
- Line 11>>> sets setName methods
- Line 12>>> calling printCustomer() and out put
- C001 - Danapala
- C002 - Gunapala
- C003 - Somapala

//

28. What will be the output when you compile and run the program?

```
class Test{
int x=10;
static int y=20;
}
class Demo{
public static void main(String args[]){
System.out.println(MyClass.x); //Line 1
System.out.println(MyClass.y); //Line 2
MyClass c1=new MyClass();
System.out.println(c1.x); //Line 3
System.out.println(c1.y); //Line 4
}
}
```

A. Compile Error at line 1
B. Compile Error at line 2
C. Compile Error at line 4
D. Compile Error at line 3

//

29. What will be the output when you compile and run the program?

```
class MyClass{
int x;
static int y;
}
class Demo{
public static void main(String args[]){
MyClass c1=new MyClass();
MyClass c2=new MyClass();
MyClass c3=new MyClass();
c1.x=1; c1.y=2; c2.x=10;
c2.y=20;
```

```
c3.x=100;
c3.y=200;
System.out.println(c1.x+" "+c1.y+" "+c2.x+"
"+c2.y+" "+c3.x+" "+c3.y);

}
}
```

A. 1 2 10 20 100 200
B. 1 1 10 10 100 100
C. 1 2 10 2 20 2
D. 1 100 10 100 100 100
E. 1 200 10 200 200 200
F None of the above answer is // 1 200 10 200 100 200

//

30. Create a class Rectangle with attributes length and width, each of which defaults to 1. Provide methods that calculate the rectangle's perimeter and area. It has set and get methods for both length and width. The set methods should verify that length and width are each floating-point numbers larger than 0.0 and less than 20.0. Write a program to test class Rectangle.

-
```
class Rectangle{
    private double length;
    private double width;
    Rectangle(){}
    Rectangle(double length,double width){
            this.length=length;
            this.width=width;
            }

    public void setLength(double length,double width){
             if((length>0.0&&width>0.0)&(length<20.0&&width<20.0)){
              this.length=length;
              this.width=width;
              getArea();
              }else{
                      System.out.println("invalid width or length");
                      }

            }
```

```
public void getArea(){
        double area=length*width;
   System.out.println("Area = "+area);
        }

}
class Main1{
public static void main(String args[]){


Rectangle r1=new Rectangle();
r1.setLength(2.0,2.0);
}
}
```

//

31. What will be the output when you compile and run the program?

```
class Demo{
static void staticMethod(){
System.out.println("staticMethod");
}
void instanceMethod(){
System.out.println("instanceMethod");
}
public static void main(String args[]){
instanceMethod(); //Line 1
staticMethod(); //Line 2
Demo.instanceMethod(); //Line 3
Demo.staticMethod(); //Line 4
Demo d1=new Demo(); //Line 5
d1.instanceMethod(); //Line 6
d1.staticMethod(); //Line 7
}
}
```

A. Compile Error at line 1
B. Compile Error at line 2
C. Compile Error at line 3
D. Compile Error at line 4
E. Compile Error at line 5
F. Compile Error at line 6

//

32. What will be the output when you compile and run the program?

```
class Test{
int x=10;
static int y=20;
Test(int i, int j){ x=i; y=j; }
void printXY(){
System.out.print(x+" "+y+" ");
}
}
class Demo{
public static void main(String args[]){
Test t1=new Test(1,2);
Test t2=new Test(10,20);
Test t3=new Test(100,200);
t1.printXY();
t2.printXY();
t3.printXY();
}
}
```

A. 1 2 10 20 100 200
B. 1 1 10 10 100 100
C. 1 2 10 2 20 2
D. 1 100 10 100 100 100
E. 1 200 10 200 200 200
F. None of the above // 1 200 10 200 100 200

//

33. Which of the following code lines are illegal?

```
class Test{
int x=10;
static int y=20;
static void staticMethod(){}
void instanceMethod(){}
void mA(){
System.out.println(x);//Line 1
System.out.println(y);//Line 2
staticMethod(); //Line 3
instanceMethod(); //Line 4
}
```

```
static void mB(){
System.out.println(x);//Line 5
System.out.println(y);//Line 6
staticMethod(); //Line 7
instanceMethod(); //Line 8
}
}
```

- error: non-static variable x cannot be referenced from a static context ● error: non-static variable x cannot be referenced from a static context

//

34. Explain difference between “static variables” and “instance variables”

- Instance variables are fields declared within a class but outside any method. They are used to store unique data for each instance of the class.
- Unlike static variables which are shared among all instances, instance variables have distinct values specific to each individual object.

//

35. Given:

```
class Demo{
int a=10;
static int b=20;
public static void main(String args[]){
int c=20;
Demo ob=new Demo();
System.out.println(a); //Line 1
System.out.println(b); //Line 2
System.out.println(c); //Line 3
System.out.println(Demo.a); //Line 4
System.out.println(Demo.b); //Line 5
System.out.println(Demo.c); //Line 6
System.out.println(ob.a); //Line 7
System.out.println(ob.b); //Line 8
System.out.println(ob.c); //Line 9
}
}
```

Which of these lines will cause compile error?

A. Compile Error at line 1
B. Compile Error at line 2
C. Compile Error at line 3
D. Compile Error at line 4
E. Compile Error at line 5
F Compile Error at line 6
G. Compile Error at line 7

H. Compile Error at line 8
I. Compile Error at line 9

//

36. Extends the class “Date” of Question 26 with this given functionalities.

```
//--------------------Demo.java-----------------------
class Demo{
public static void main(String args[]){
Date d1=new Date();
d1.set(Date.YEAR,2016); //set(int field, int value)
d1.set(Date.MONTH,5);
d1.set(Date.DAY,30);
d1.printDate(); //2016-5-30
}
}
```

//

37. Given:

```
class Demo{
static int m(int i, String s) {
System.out.print("["+s+","+i+"]"+",");
return i;
}
public static void main (String[] args) {
int i = 0;
int a[] = new int[3];
a[m(i++,"a")] = m(i++,"b");
System.out.print(a[0]+","+a[1]+","+a[2] +","+ i);
}
}
```

What is the output?

A. Prints: [a,0],[b,1],1,0,0,2
B. Prints: [a,1],[b,2],0,2,0,2
C. Prints: [a,1],[b,0],0,0,0,2
D. Prints: [a,2],[b,1],0,0,1,2
E. Runtime error
F Compiler error

//

38. Given:

```
class Demo{
public static int m(int i) {
```

```
System.out.print(i + ", ");
return i;
}
public static void main(String s[]) {
int i=0;
int j = m(++i) + m(++i) * m(++i) %m(++i) + m(++i);
System.out.print( j % 5);
}
}
```

What is the result of attempting to compile and run the program?

A. Prints: 1,2,3,4,5,1
B. Prints: 1,2,3,4,5,2
C. Prints: 1,2,3,4,5,3
D. Prints: 1,2,3,4,5,4
E. Prints: 1,2,3,4,5,5
F. Compiler error

//

39. Explain difference between “static methods ” and “instance methods”

- Instance variables are fields declared within a class but outside any method. They are used to store unique data for each instance of the class.
- Unlike static variables which are shared among all instances, instance variables have distinct values specific to each individual object.

//

40. Write a java class called “Cylinder” with attributes length (type double) and radius (type double) of the Cylinder.

a. Create methods that calculate the volume and area of the Cylinder.
b. Implements two constructors default constructor and parameterized constructor.
c. Implements setters and getters of each attributes of the class cylinder.

Write an application to test your class “Cylinder

- class Cylinder{
- double length;
- double radius;
- Cylinder() {
-
- }
- Cylinder(double length, double radius) {

```
● this.length = length;
● this.radius = radius;
● }
●
● public double getLength() {
● return length;
● }
● public void setLength(double length) {
● this.length = length;
●
● }
● public double getRadius() {
● return radius;
● }
● public void setRadius(double radius) {
●
● this.radius = radius;
● }
● public double calculateVolume() {
● double f=3.14;
● double volume=f*(radius * radius)*length;
● return volume;
●
● }
● public double calculateArea() {
● double f=3.14;
● double area=2*f * radius;
● return area;
● }
● }
● class Main1{
● public static void main(String[] args) {
●
● Cylinder c1 = new Cylinder(2.0, 3.0);
●
●
● System.out.println("Volume: " + c1.calculateVolume()); ●
System.out.println("Area: " + c1.calculateArea());
● }
● }
```

41. What is the output, and explain the answer.

```
class Box{
int length;
int width;
int height;
static{
System.out.println("Box is loaded into memory");
}
}
class Demo{
public static void main(String args[]){
Box b1=new Box();
}
}
```

- Output >>> Box is loaded into memory

//

42. What is the output, and explain the answer.

```
class Box{
int length;
int width;
int height;
{
System.out.println("A box object is created..");
}
}
class Demo{
public static void main(String args[]){
Box b1=new Box();
Box b2=new Box();
}
}
```

- A box object is created..
- A box object is created..

//

43. What is the output, and explain the answer.

```
class Box{
int length;
int width;
int height;
static{
System.out.println("Box is loaded into memory");
```

```
}
{
System.out.println("A box object is created..");
}
}
class Demo{
public static void main(String args[]){
Box b1=new Box();
Box b2=new Box();
Box b3=new Box();
}
}
```

- Box is loaded into memory
- A box object is created..
- A box object is created..
- A box object is created..

//

44. Given:

```
class MyClass{
int x=10;
static int y=20;
MyClass(){
System.out.print("3 ");
}
static{
System.out.print("1 ");
}
{
System.out.print("2 ");
}
static void mB(){
System.out.print("4 ");
}
}
class Demo{
public static void main(String args[]){
//Insert code here line 12
}
}
```

Write the output generated by the given program inserting the following code fragments at line 12.

A. new MyClass(); >>> 1 2 3
B. new MyClass(); >>> 1 2 3 2 3
new MyClass();
C. MyClass.mB(); >>> 1 4
D. new MyClass().mA();>>> error
E. MyClass.mB(); >>> 1 4
F. MyClass c1; >>> compile and run

//

45. Given:
class Test{
int x=10;
static int y=20;
void mA(){
System.out.println(this.x);//Line 1
System.out.println(this.y);//Line 2
}
static void mB(){
System.out.println(this.x);//Line 3
System.out.println(this.y);//Line 4
}
}
What is the result of attempting to compile and run the
above program?
A. Compile error at Line 1
B. Compile error at Line 2
C. Compile error at Line 3
D. Compile error at Line 4
F. Program will runs without errors

//

46. Describe “Constructor overloading” and “Method
overloading” with examples.

- While constructor overloading involves defining multiple constructors in class.
- public class Student {
- private String name;
- private int age;
-
- public Student() {
- this.name = "Unknown";
- this.age = 0;
- }
-

```
● public Student(String name) {
● this.name = name;
● this.age = 0;
● }
●
● public Student(String name, int age) {
● this.name = name;
● this.age = age;
● }
●
● public static void main(String[] args) {
● Student student1 = new Student();
● Student student2 = new Student("John"); parameter
● Student student3 = new Student("Alice", 25);
● }
● }
```

● method overloading involves defining multiple methods with the same name but different parameters.

```
● public class Calculator {
● public int add(int a, int b) {
● return a + b;
● }
●
● public int add(int a, int b, int c) {
● return a + b + c;
● }
●
● public double add(double a, double b) {
● return a + b;
● }
● public static void main(String[] args) {
● Calculator calculator = new Calculator();
●
● int result1 = calculator.add(2, 3);
                int result2 = calculator.add(1, 5, 7);

● double result3 = calculator.add(2.5, 3.5);
● }
● }
●
```

47. Which of the following lines of code could be inserted at line 12 and still allow the code to compile? Explain your answers.

```
class Example{
public static void myMethod(int y,double x){
System.out.println("myMethod(int,double)");
}
public static void main(String args[]){
byte b=10;
short s=20;
int x=30;
long y=10;
float f=1.0f;
double d=1.2;
char ch='A';
//Insert code here //Line 12
}
}
```

A. myMethod(b, b); >>> myMethod(int,double)
B. myMethod(s, s);>>> myMethod(int,double)
C. myMethod(b, s); >>> myMethod(int,double)
D. myMethod(s, b);>>> myMethod(int,double)
E. myMethod(x, x); >>> myMethod(int,double)
F. myMethod(x, y);>>> myMethod(int,double)
G. myMethod(y, x); >>> error
H. myMethod(x, b);>>> myMethod(int,double)
I. myMethod(y, y); >>> error
J. myMethod(x, f);>>> myMethod(int,double)
K. myMethod(x, d); >>> myMethod(int,double)
L. myMethod(ch,ch);>>> myMethod(int,double)
M.myMethod(x,ch); >>> myMethod(int,double)
N. myMethod(ch,d);>>> myMethod(int,double)

//

48. Which of the following lines of code could be inserted at line 3 and still allow the code to compile? Explain your answers.

```
class Example{
public static void myMethod(int x,double y){}
//Line 3
}
```

A. public static void myMethod(char x,double y){}

- This will not compile because the parameter types do not match the original method

B. public static void myMethod(double x,char y){}

- This will not compile because the parameter types do not match the original method

C. public static void myMethod(int a,double b){}

- This will compile. The parameter names can be different.

D. public void myMethod(int x,double y){}

- This will not compile because the return type does not match the original method.

E. public static void myMethod(byte x,double y){}

- This will not compile because the parameter types do not match the original method.

F. public static void myMethod(short x,double y){}

- This will not compile because the parameter types do not match the original method.

G. public static void myMethod(float x,double y){}

- This will not compile because the parameter types do not match the original method.

H. public static void myMethod(double x,double y){}

- This will not compile because the parameter types do not match the original method I.

public static int myMethod(int x,double y){
return 100;}

- This will not compile because the return type does not match the original method.

// 49.

Consider this class example:

```
class MyPoint {
void myMethod() {
int x, y;
x = 5;
y = 3;
System.out.print( " ( " + x + ", " + y + " ) " );
switchCoords( x, y );
System.out.print( " ( " + x + ", " + y + " ) " );
}
void switchCoords( int x, int y ) {
int temp;
temp = x;
x = y;
y = temp;
System.out.print( " ( " + x + ", " + y + " ) " );
}
}
```

What is printed to standard output if myMethod() is executed?

A. (5, 3) (5, 3) (5, 3)
B. (5, 3) (3, 5) (3, 5)
C. (5, 3) (3, 5) (5, 3)
D. (3, 3) (5, 5) (5, 3)

//

50. Describe “Method Call by Values” vs “Method Call by Reference” with appropriate examples

- call by value, the actual value of the argument is passed to the method. This means that any changes made to the parameter inside the method do not affect the original value outside the method.

- primitive data types (int, float, char, etc.) are examples of call by value.

—---------- ● While Java strictly uses call by value, the confusion arises when dealing with objects. When an object is passed as an argument to a method, the reference to the object is passed by value.

- public class CallByReferenceExample {
- public static void main(String[] args) {
- StringBuilder str = new StringBuilder("Hello");
- System.out.println("Before method call: " + str);
- modifyReference(str);
- System.out.println("After method call: " + str);
- }
-
- static void modifyReference(StringBuilder s) {
- s.append(" World");
- System.out.println("Inside method: " + s);
- }
- }

//

51. Which of the following lines are illegal? Explain your answer.

```
class Cat{
private String name;
}
class Demo{
public static void main(String args[]){
Cat cat; //Line 1
Cat[] cats; //Line 2
cat=new Cat[5]; //Line 3
cat=new Cat(); //Line 4
cats=new Cat(); //Line 5
cats=new Cat[5]; //Line 6
}
}
```

- Line 4 >>>illegal

- Line 5 >>> illegal
- cannot assign a single Cat object to an array reference variable.

//

52. Insert code at line50 to get the output as follow

```
//[Aldo, Bear, Toby, Teddy, Henry]
class Cat{
private String name;
Cat(String name){this.name=name;}
public String getName(){return name;}
public void setName(String name){this.name=name;}
}
class Demo{
public static void main(String args[]){
Cat[] cats={new Cat("Aldo"),
new Cat("Bear"),
new Cat("Toby"),
new Cat("Teddy"),
new Cat("Henry")};
//Line 50
}
}
```

- class Cat {
- private String name;
-
- Cat(String name) {
- this.name = name;
- }
-
- public String getName() {
- return name;
- }
-
- public void setName(String name) {
- this.name = name;
- }
-
- public String toString() {
- return name;
- }
- }
- class Main1 {
- public static void main(String args[]) {
- Cat[] cats = {

```
● new Cat("Aldo"),
● new Cat("Bear"),
● new Cat("Toby"),
● new Cat("Teddy"),
● new Cat("Henry")
● };
●
● StringBuilder output = new StringBuilder("[");
● for (Cat cat : cats) {
● output.append(cat).append(", ");
● }
●
● if (cats.length > 0) {
● output.setLength(output.length() - 2);
● }
●
● output.append("]");
● System.out.print(output);
● }
● }
```

//

53. Create a class called Employee that includes three instance variables a first name (type String), a last name (type String) and a monthly salary (double). Provide a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, do not set its value. Write a test application named EmployeeTest that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10% raise and display each Employee's yearly salary Again.

//

54. Given:

```
class Item{
 private int code;
 Item(int code){
this.code=code;
}
```

```
void setCode(Item item){
this.code=item.code;
}
void printCode(){
System.out.println("code : "+code);
}
}
class Demo{
public static void main(String args[]){
Item item1=new Item(1001);
item1.printCode();
Item item2=new Item(2002);
item2.setCode(item1);
item2.printCode();
}
}
```

What is the result of attempting to compile and run the above program?

- code : 1001
- code : 1001

//

56. Given the following code, what will be the output?

```
class ValHold{
public int i = 10;
}
class ObParm{
public static void main(String argv[]){
ObParm o = new ObParm();
o.amethod();
}
public void amethod(){
int i = 99;
ValHold v = new ValHold();
v.i=30;
another(v,i);
System.out.println(v.i);
}//End of a method
public void another(ValHold v, int i){
i=0;
v.i = 20;
ValHold vh = new ValHold();
```

```
v = vh;
System.out.println(v.i+ " "+i);
}//End of another
}
```

A. 10,0, 30
B. 20,0,30
C. 20,99,30
C. 10,0,20

//

57. Create a class called “Invoice” that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables—a part number (type String), a part description (type String), a quantity of the item being purchased (type int) and a price per item (double). Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named getInvoiceAmount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named InvoiceTest that demonstrates class Invoice’s capabilities.

//

58. Complete the class Box with given functionalities.

```
class Box{
private int length;
private int width;
private int height;
//Insert code to complete the class “Box”
}
class Demo{
public static void main(String args[]){
Box b1=new Box();
b1.setLength(12);
b1.setWidth(5);
b1.setHeight(3);
b1.printVolume();
b1.setDimension(120,50,30);
```

```
System.out.println("Volume "+b1.getVolume());
Box b2=new Box(4,2,3);
b2.printVolume();
Box b3=new Box(b2);
//copy dimensions of b2 into b3
b3.printVolume();
Box b4=new Box(10);//length for a square cube
b4.printVolume(); //
Box b5=new Box();
b5.setDimension(12); //length for a square cube
b5.printVolume();
Box b6=new Box();
b6.printVolume();
b6.setDimension(b1);
//copy dimensions of b1 into b6
b6.printVolume();
Box b7=b3.getCopy();
b7.printVolume();
}
}
```

- class Box {
- private int length;
- private int width;
- private int height;
-
- public Box() {}
-
- public Box(int length, int width, int height) { ● this.length = length;
- this.width = width;
- this.height = height;
- }
-
- public Box(Box otherBox) {
- this.length = otherBox.length;
- this.width = otherBox.width;
- this.height = otherBox.height;
- }
-
- public void setLength(int length) {
- this.length = length;
- }
-

● public void setWidth(int width) {
● this.width = width;
● }
●
● public void setHeight(int height) {
● this.height = height;
● }
●
● public void setDimension(int length, int width, int height) { ●
this.length = length;
● this.width = width;
● this.height = height;
● }
●
● public void setDimension(int side) {
● this.length = side;
● this.width = side;
● this.height = side;
● }
●
● public void setDimension(Box otherBox) { ●
this.length = otherBox.length;
● this.width = otherBox.width;
● this.height = otherBox.height;
● }
●
● public void printVolume() {
● int volume = length * width * height;
● System.out.println("Volume: " + volume);
● }
●
● public int getVolume() {
● return length * width * height;
● }
●
● public Box getCopy() {
● return new Box(this);
● }
● }
●
● class Main1 {
● public static void main(String args[]){
● Box b1=new Box();

- b1.setLength(12);
- b1.setWidth(5);
- b1.setHeight(3);
- b1.printVolume();
- b1.setDimension(120,50,30);
- System.out.println("Volume "+b1.getVolume());
- Box b2=new Box(4,2,3);
- b2.printVolume();
- Box b3=new Box(b2);
- //copy dimensions of b2 into b3
- b3.printVolume();
- Box b4=new Box(10);//length for a square cube
- b4.printVolume(); //
- Box b5=new Box();
- b5.setDimension(12); //length for a square cube
- b5.printVolume();
- Box b6=new Box();
- b6.printVolume();
- b6.setDimension(b1);
- //copy dimensions of b1 into b6
- b6.printVolume();
- Box b7=b3.getCopy();
- b7.printVolume();
- }
- }

//

59. Which of the following lines of code could be inserted at line 12 and still allow the code to compile? Explain your answers.

```
class Example{
public static void myMethod(int y,double x){
System.out.println("myMethod(int,double)");
}
public static void myMethod(double x,int y){
System.out.println("myMethod(double,int)");
}
public static void main(String args[]){
byte b=10;
short s=20;
int x=30;
long y=10;
```

```
float f=1.0f;
double d=1.2;
char ch='A';
//Insert code here
}
}
```

A. myMethod(b, b); >>> error
B. myMethod(s, s);>>> error
C. myMethod(b, s); >>> error
D. myMethod(s, b);>>> error
E. myMethod(x, x); >>> error
F. myMethod(x, y);>>>myMethod(int,double)
G. myMethod(y, x); >>>myMethod(double,int)
H. myMethod(x, b);>>>error
I. myMethod(y, y); >>>error
J. myMethod(x, f);>>>myMethod(int,double)
K. myMethod(x, d); >>>myMethod(int,double)
L. myMethod(ch,ch);>>>error
M.myMethod(x,ch); >>>error
N. myMethod(ch,d);>>>myMethod(int,double)

//

60. Given:

```
class M {
public static int m(int i) {
System.out.print(i + ", ");
return i;
}
public static void main(String s[]) {
m(m(1) + m(2) % m(3) * m(4));
}
}
```

What is the result of attempting to compile and run the program?

A. Prints: 1, 2, 3, 4, 0,
B. Prints: 1, 2, 3, 4, 12,
C. Prints: 1, 2, 3, 4, 3,
D. Prints: 2, 3, 4, 1, 9,
E. Prints: 1, 2, 3, 4, 9,
F. Prints: 2, 3, 4, 1, 3,
G. Compiler error
H. None of the above

//

61. What will be the result when you compile and the following program?

```
class MyClass{
 MyClass(){
System.out.println("MyClass()");
}
 MyClass(int i){
this(i,i);
System.out.println("MyClass(int)");
}
 MyClass(int i,int j){
this();
System.out.println("MyClass(int,int)");
}
}
class Demo{
 public static void main(String args[]){
 MyClass c1=new MyClass(100);
}
}
```

- MyClass()
- MyClass(int,int)
- MyClass(int)

//

62. Which of the following lines of code could be inserted at line 30 and still allow the code to compile?

```
class MyClass{
int x=10;
static int y=20;
 MyClass(double i){
System.out.println("MyClass(double)");
}
 MyClass(int i){
System.out.println("MyClass(int)");
}
 MyClass(int i,double j){
//Insert code here //Line 30
System.out.println("MyClass(int,double)");
}
```

}

A. this();

B. this(j);

C. this(i)

D. this(100);

E. this(1.5f);

F. this(i, j);

G. this(100L);

H. this(x);

I. this(y);