INSTITUTE OF SOFTWARE ENGINEERING

GRADUATE DIPLOMA IN SOFTWARE ENGINEERING

ASSIGNMENT NAME

Object Oriented Programming

ASSIGNMENT NO

01

NUMBER OF QUESTIONS: 62

NUMBER OF COMPLETED QUESTIONS: 62

NUMBER OF REMAINING QUESTIONS: 00

STUDENT NAME: Maheeshi Jayarathna

NIC: 200062000066

BATCH NO: 61

01.

a. Class/Template

A Class is a blueprint or a copy of attributes and behaviours that an object can have.

ex → “Person” as a class then his name/birthday/career/address are the attributes and behaviours that person can have

b. Object/ Instance

Any living thing or nonliving thing which can describe is called an object. Each object has attributes and behaviours.

ex → take ‘animals’ as a class then rabbit/deer/tiger/fox…etc are the objects of thant class named ‘animals’

c. Methods/ Functions

A method is a set of code which is used to perform a certain action

d. Attributes/ Properties

Attributes are the information about an object. Describe the characteristics about an object.

e. Reference Variables

A reference variable is a variable that points to an object of a given class, letting you access the value of an object. A reference variable does not store its own values.

f. Primitive variables

g. Method parameters

Information can be passed to methods as a parameter. The parameters in the method act as variables.Parameters are specified after the method name in parentheses. We can add as many parameters as we want, separating them with commas.

h. Local variables

i. Default values

j. Declaration values

02.

- D

03.

- A

04.

- Attributes are things an object has and methods are things the object can do
  Attributes - name/age/address
  Method - play/sing/run

05.

- 2/3/4 - variables related to “class Box” has not declared inside the “class Demo”

06.

- Volume : 0
  length of box : 0
  width of box : 0
  height of box : 0

  The default value of instance variable of int type is 0

07.

- Volume : 180
  length of box : 12
  width of box : 5
  height of box : 3

Since values are assigned to the variables, the volume is created and printed when the printVolume method is first called. After that, the values are assigned and printed when output is printed.

08.

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

The constructor is called when the object is created, so it is printed as the default constructor first. Then values are initialised to the variables.

09.

- Parameterized constructor
  Volume : 60
  Parameterized constructor
  Volume : 180

When the constructor is called in the first object, sout is printed. When the printVolume method is called, the values are initialized and the volume is printed. The same happens with the second object.

10.

- compile error

Private variables can be accessed only in the same classes. If you want to access a private variable, you must create a method in that class and access it.

11.

- Line 4
  Line 5
  Line 6

A variable is not initialised within an instance block.

12.

- Constructor is a block of code similar to the method. It is called when an instance of the class is created.constructors are used to assign values to the class variables at the time of object creation.Each time an object is created using a new() keyword, the default constructor is called to assign initial values to the data members of the same class.

```
class Car{
    int year;
    String name;
    String colour;

    public Car(){
    }
}
```

13.

- Constructor is used to initialize an object whereas method is used to exhibits functionality of an object.
- Constructors are invoked implicitly whereas methods are invoked explicitly.
- Constructor does not return any value where the method may/may not return a value.
- In case constructor is not present, a default constructor is provided by java compiler. In the case of a method, no default method is provided.
- Constructor should be of the same name as that of class. Method name should not be of the same name as that of class.

14.

- A , B , C

15.

- Line 2,3
  The variable is not declared.

16.

- C

17.

- A , B , D

18.

- 100 101
  0 0
  The argument used in the first sout is print. In the second sout, because the two objects are assigned together, '0' printed.

19.

- Code : 3001
  3001 is printed because printCode is called using argument.

20.

- Code : 2001
  Because this keyword is used, values are printed in the instance block.

21.

- B , C , D

22.

- A , D , E

23.

- Encapsulation means making attributes private and accessing data through methods.

```
class Student{
      private String name;
      private int age;
      private double marks;

      public void setName{
           name=x;
      }
      public void setAge{
           age=y;
      }
      public void setMarks{
           marks=z;
      }
      public String getName{
           return name;
      }
      public int getAge{
           return age;
      }
      public double getMarks{
           return marks;
      }
  }
```

24.

- Tightly encapsulation -

  All attributes are made private and data access is done by getters and setters.

```
class Student {
     private String name;
     private int age;

     public void setName(){
```

```
                        name=x;
                    }

                    public void setAge(){
                        age=y;
                    }

                    public String getName(){
                         return name;
                    }

                     public int getAge(){
                         return Age;
                    }

                }
```

- Loosely encapsulation -
  One or more attributes are made private and data access is done by ළු getters and setters.

```
            class Student {
                    private String name;
                    int age;

                    public void setName(){
                        name=x;
                    }

                    public String getName(){
                         return name;
                    }
                }
```

25.

- D

26.

```
//--------------------Date.java-----------------------
class Date{
   int year=1970;
   int month=1;
   int day=1;
   void printDate(){
       System.out.println(year+"-"+month+"-"+day);
   }
   void setYear(int y){
       year=y;
   }
   void setMonth(int m){
       month=m;
   }
   void setDay(int d){
       day=d;
   }
   int getYear(){
       return year;
   }
   int getMonth(){
       return month;
   }
   int getDay(){
       return day;
   }
}

//--------------------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());
   }
}
```

27.

28.
- A , B , C , D

29.
- F

30.

```
class Rectangle{
   private double length;
   private double width;
   {
       length=1;
       width=1;
   }
   void set(double length,double width){
       if(0<length && length<20 && 0<width && width<20){
           this.length=length;
           this.width=width;
       }
   }
   double getLength(){
       return length;
   }
   double getWidth(){
       return width;
   }
   void perimeter(){
       double perimeter=2*(length+width);
       System.out.println("perimeter :- "+perimeter);
   }
   void area(){
       double area=length*width;
       System.out.println("area :- "+area);
   }

}
class Demo{
   public static void main(String args[]){
       Rectangle r1=new Rectangle();
       r1.set(15,15);
```

```
        System.out.println("Length :- "+r1.getLength());
        System.out.println("Width :- "+r1.getWidth());
        r1.perimeter();
        r1.area();
    }
}
```

31.

- A , C

32.

- F

33.

- Line 5 , 8

34.

Instance variables	Static (class) variables
Instance variables are declared in a class, but outside a method, constructor or any block.	Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.	Static variables are created when the program starts and destroyed when the program stops.
Instance variables can be accessed directly by calling the variable name inside the class. However, within static methods (when instance variables are given accessibility), they should be called using the fully qualified name. *ObjectReference.VariableName*.	Static variables can be accessed by calling with the class name *ClassName.VariableName*.

Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.	There would only be one copy of each class variable per class, regardless of how many objects are created from it.

Example

```
public class Demo{
   int box;
   static int data = 30;

   public static void main(String args[]){
          Demo d1 = new Demo();

          System.out.println("Value of instance variable: "+b1.box);
          System.out.println("Value of static variable: "+Demo.data);
   }
}
```

35.

- A , D , F , I

36.

```
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
    }
}
class Date {

    static final int YEAR = 3;
    static final int MONTH = 6;
    static final int DAY = 9;

    int year;
    int month;
    int day;

    public void set(int field, int value) {
        if (field == YEAR)
            this.year = value;
        else if (field == MONTH)
            this.month = value;
        else if (field == DAY)
            this.day = value;
    }

    public void printDate() {
        System.out.println(year + "-" + month + "-" + day);
    }
}
```

37.

- A

38.

- C

39.

- Instance methods are methods which require an object of its class to be created before it can be called.

- Static methods are the methods in Java that can be called without creating an object of class

40.

41.

- Output - “Box is loaded into memory”

There is a System out called “Box is loaded into memory” in the static method in class Box.

42.

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

When creating the sout object in the instance block, it is printed.

43.

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

The sout of the static block is printed when that class is loaded into the ram. Then, when creating the sout object in the instance block, it is printed.

44.

- A. 1 2 3
  B. 1 2 3 2 3
  C. 1 4
  D. Compile error
  E. 1 4
  F. No output

45.

- C , D

46.

- Constructor overloading -

  Constructor overloading is using more than one constructor in an instance class
- Method overloading -

  Method overloading is creating two or more methods that have the same name if they have different numbers of parameters or different types of parameters.

47.

- A , B , C , D , E , F , H , J , K , L , M , N

48.

- A , B , E , F , G , H

49.

- B

50.

- Call By Value:
  In this parameter passing method, values of actual parameters are copied to function's formal parameters and the two types of parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of the caller.

- Call by Reference:
  Both the actual and formal parameters refer to the same locations, so any changes made inside the function are actually reflected in actual parameters of the caller.

51.

- Line 3
  Line 5

52.

```
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
                System.out.print("[");
                for(Cat c1:cats){
                        System.out.print(c1.getName()+",");
                }
                System.out.println(" \b\b]");
        }
}
```

53.

```
 class Employee{
        String firstName;
        String lastName;
        double monthlySalary;

        Employee(String firstName,String lastName,double monthlySalary){
                this.firstName= firstName;
                this.lastName= lastName;
                this.monthlySalary= monthlySalary;
        }



        String getfirstName(){
                return firstName;
        }

        String getlastName(){
                return lastName;
        }

        double getmonthlySalary(){
                return monthlySalary;
        }

        double getYearSalary(){
                 double yearSalary=monthlySalary*12;
                 return yearSalary;
        }

        double getfinalSalary(){
                double finalSalary=monthlySalary*12*1.1;
                return finalSalary;
        }
}


class Demo{
        public static void main(String args[]){
                Employee e1=new Employee("Maheeshi","Jayarathna",200000);
                Employee e2=new Employee("Shamodha","Rathnamalala",500000);


                System.out.println("first Name :" + e1.getfirstName());
                System.out.println("last Name :" + e1.getlastName());
                System.out.println("final  Salary:" + e1.getfinalSalary());

                System.out.println();
                System.out.println("first Name :" + e2.getfirstName());
                System.out.println("last Name :" + e2.getlastName());
                System.out.println("final  Salary:" + e2.getfinalSalary());
                }
        }
```

54.

-
  ```
  code : 1001
  code : 1001
  ```

55.

```
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,05);
        d1.set(Date.DAY,30);
        d1.printDate(); //2016-5-30
        Date d2=new Date(2016,1,31);
        d2.printDate(); //2016-1-31
        Date d3=new Date(d2);
        d3.printDate(); //2016-1-31
        Date d4=new Date();
        d4.set(d1);
        d4.printDate();
        Date d5=Date.getDateInstance();
        d5.printDate(); //1970-1-1
    }
}
class Date {
    static final int YEAR = 3;
    static final int MONTH = 6;
    static final int DAY = 9;
    int year=1970;
    int month=1;
    int day=1;
    Date() {
    }
    Date(Date d2) {
        this.year= d2.year;
        this.month=d2.month;
        this.day= d2.day;
    }
    Date(int year, int month, int day) {
        this.year=year;
        this.month=month;
        this.day=day;
    }
    static Date getDateInstance() {
        Date d5=new Date();
        return d5;
    }
    void set(int field, int value) {
        if (field == YEAR)
            this.year = value;
        else if (field == MONTH)
            this.month = value;
        else if (field == DAY)
            this.day = value;
```

```
  }

  void printDate() {
     System.out.println(year + "-" + month + "-" + day);
  }

  void set(Date d1) {
     this.year=d1.year;
     this.month=d1.month;
     this.day=d1.day;
  }
}
```

56.

- D

57.

```
class Demo{
	public static void main(String args[]){
		Invoice v1 = new Invoice("S001","toy",5,65.0);
		v1.getNumer();
		v1.getDescription();
		v1.getQuantity();
		v1.getItemPrice();
		v1.getInvoiceAmount();

		System.out.println();
		Invoice v2 = new Invoice("S002","pen",-1,-100.0);
		v2.getNumer();
		v2.getDescription();
		v2.getQuantity();
		v2.getItemPrice();
		v2.getInvoiceAmount();
	}
}

class Invoice{
	private String number;
	private String description;
	private int quantity;
	private double itemPrice;

	Invoice(String number,String description,int quantity,double itemPrice){
		this.number=number;
		this.description=description;
		this.quantity=quantity;
		this.itemPrice=itemPrice;
	}

	public void setNumber(String number){
		this.number=number;
	}

	public void setDescription(String description){
		this.description=description;
	}

	public void setQuantity(int quantity){
			this.quantity=quantity;
	}

	public void setItemPrice(double itemPrice){
		this.itemPrice=itemPrice;
	}

	public void getNumer(){
		System.out.println("Enter product Number : "+number);
	}

	public void getDescription(){
		System.out.println("Enter product Description : "+description);
	}

	public void getQuantity(){
		System.out.println("Enter Item Quanity  : "+quantity);
	}

	public void getItemPrice(){
		System.out.println("Enter Itme Price : "+itemPrice);
	}

	public void getInvoiceAmount(){
		double amount=0.0;
		if(quantity>0 || itemPrice>0.0){
		amount=quantity*itemPrice;
	}
	System.out.println("Total Price : "+amount);
```

```
  }

}
```

58.

```
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;
  Box(){}
  Box(int length, int width, int height) {//
     this.length=length;
     this.width=width;
     this.height=height;
  }
```

```
public Box(Box b2) {//
    this.length=b2.length;
    this.width=b2.width;
    this.height=b2.height;
}
public Box(int length) {//
    this.length=length;
    this.width=length;
    this.height=length;
}
void setLength(int length) {//
    this.length=length;
}
void setWidth(int width){//
    this.width=width;
}
void setHeight(int height){//
    this.height=height;
}
void printVolume() {//
    System.out.println("Volume is : "+(length*width*height));
}
void setDimension(int length, int width, int height) {//
    this.length=length;
    this.width=width;
    this.height=height;
}
int getVolume() {//
    return (length*width*height);
}
void setDimension(int length){//
    this.length=length;
    this.width=length;
    this.height=length;
}
void setDimension(Box b1) {//
    this.length=b1.length;
    this.width= b1.width;
    this.height=b1.height;
}


Box getCopy() {
    return new Box(length,width,height);
```

```
  }
}
```

59.

- F , G , J , K , N

60.

- E

61.

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

62.

- B , C , D , E , G , I