**EXERCISE 1**

**Question1**

**Class Diagram**

**A screenshot of a social media post

Description automatically generated**

**Output**

A screenshot of a cell phone

Description automatically generated

**Why such output is printed?**

Answer: Dictionary is a derived class that inherits instance variables from the base class. Therefore, the variable pages can be access in class Dictionary to perform the calculation of the number of definitions per page

**Question2**

1. **Class Diagram**

**A screenshot of a cell phone

Description automatically generated**

ii) when a circle object is created, what are its public members?

Answer: radius

iii) What members of the Shape class are not accessible to the circle class methods?

Answer: area

iv) c.setRadius(10.0); **LEGAL**

s.setRadius(10.0); **ILLEGAL**

system.out.println(c.getArea()); **LEGAL**

System.out.println(s.getArea()); **LEGAL**

c.setRadius(10.0) **LEGAL**

**Question5**

Answer:

class Rectangle

{

private double width;

private double length;

Rectangle(double \_w, double \_i)

{

width=\_w;

length=\_i;

}

protected double calculateArea()

{

return width\*length;

}

public void display()

{

System.out.println("Width : "+width);

System.out.println("Length : "+length);

}

}

class Square extends Rectangle

{

double height;

public Square()

{

super(10.0,15.0);

height=10;

}

public double calculateVolume()

{

return height\*super.calculateArea();

}

public void display()

{

super.display();

System.out.println("height : "+height);

System.out.println("volume : "+calculateVolume());

}

}

public class TestShape2

{

public static void main(String[] args)

{

Square squarebox=new Square();

squarebox.display();

}

}

class Square extends Rectangle

{

double height;

public Square()

{

super(10.0,15.0);

height=10;

}

public double calculateVolume()

{

return height\*super.calculateArea();

}

public void display()

{

super.display();

System.out.println("height : "+height);

System.out.println("volume : "+calculateVolume());

}

}

public class TestShape2

{

public static void main(String[] args)

{

Square squarebox=new Square();

squarebox.display();

}

}

**Output**:

A screenshot of a cell phone

Description automatically generated

**EXERCISE 2**

**Question 1**

1. **Class Diagram**

**A screenshot of a cell phone

Description automatically generated**

Why such output is printed?

Answer: System.out.println(e.f)//the value 3 is pass to constructor in classX and function f() is called therefore the value carried by constructor is added by 1

**Question 2**

A close up of text on a white background

Description automatically generated

Error: super keyword is not in the first line of class B constructor

**Corrected Syntax**

public class B()

{

Super(40);

Int init=10;  
}

**Question 3**

A close up of text on a white background

Description automatically generated

Question: What is wrong with the following subclass definition?

Answer: Nothing is wrong

**Question 4**

Answer: The program cannot be compiled

Reason: class car’s constructor is invalid because it just uses one parameter which is p while c is not used.

Answer:

class Vehicle

{

private double cost;

public Vehicle(double c)

{

cost=c;

System.out.println("Cost = RM "+cost);

}

}

class Car extends Vehicle

{

private int passengers;

public Car(double c, int p)

{

super(c);

passengers=p;

System.out.println("Number of passengers : "+passengers);

}

}

public class TestVehicle

{

public static void main(String[] args)

{

Car car1=new Car(45000,4);

}

}

super(c);

passengers=p;

System.out.println("Number of passengers : "+passengers);

}

}

public class TestVehicle

{

public static void main(String[] args)

{

Car car1=new Car(45000,4);

}

}

**Output**  A picture containing drawing

Description automatically generated

**Question 5**

1. **Class Diagram**

**A screenshot of a cell phone

Description automatically generated**

ii)

A b1=new B(); **No Error**

A b1=new A(); **ERROR (object b1 is already declared. Use difference reference variable)**

B b1=new B(“Thank You”); **ERROR (object b1 is already declared. Use difference reference variable)**

B b1=new b(“Thank You”, ”Come again”); **ERROR (object b1 is already declared. Use difference reference variable)**

**EXERCISE 3**

1. Class kelasB

|  |  |
| --- | --- |
| Member | Accessible/Not Accessible |
| ab | **Accessible** |
| Xy | **Accessible** |
| Yz | **Not Accessible** |
| Var3 | **Accessible** |
| Var4 | **Accessible** |
| Var5 | **Not Accessible** |
| Var6 | **Accessible** |
| Var7 | **Not Accessible** |
| Var8 | **Not Accessible** |
| varA | **Accessible** |
| varB | **Not Accessible** |
| funR() | **Accessible** |
| funA() | **Accessible** |
| Fun6() | **Accessible** |
| Fun1() | **Accessible** |

1. Class kelasC

|  |  |
| --- | --- |
| Member | Accessible/Not Accessible |
| ab | **Accessible** |
| Xy | **Accessible** |
| Yz | **Not Accessible** |
| Var1 | **Accessible** |
| Var2 | **Accessible** |
| Var6 | **Accessible** |
| Var7 | **Not Accessible** |
| Var8 | **Not Accessible** |
| varA | **Accessible** |
| varB | **Not Accessible** |
| funR() | **Accessible** |
| funA() | **Accessible** |
| Fun2() | **Accessible** |
| Fun1() | **Accessible** |
| Fun3() | **Accessible** |

1. Class kelasD

|  |  |
| --- | --- |
| Member | Accessible/Not Accessible |
| ab | **Accessible** |
| Xy | **Accessible** |
| Yz | **Not Accessible** |
| Var1 | **Not Accessible** |
| Var2 | **Not Accessible** |
| Var3 | **Not Accessible** |
| Var4 | **Accessible** |
| Var5 | **Not Accessible** |
| Var8 | **Not Accessible** |
| varA | **Accessible** |
| VarB | **Not Accessible** |
| funR() | **Accessible** |
| Fun6() | **Accessible** |
| Fun2() | **Accessible** |
| Fun1() | **Not Accessible** |
| Fun3() | **Not Accessible** |

1. Class kelasE

|  |  |
| --- | --- |
| Member | Accessible/Not Accessible |
| ab | **Not Accessible** |
| Xy | **Accessible** |
| Yz | **Not Accessible** |
| Var1 | **Not Accessible** |
| Var2 | **Not Accessible** |
| Var3 | **Not Accessible** |
| Var4 | **Accessible** |
| Var5 | **Not Accessible** |
| Var6 | **Accessible** |
| Var7 | **Not Accessible** |
| funA() | **Accessible** |
| Fun6() | **Accessible** |
| Fun2() | **Accessible** |
| Fun1() | **Not Accessible** |
| Fun3() | **Not Accessible** |

**EXERCISE 4**

**Question2**

**Output**

**A close up of a black background

Description automatically generated**

**Why such output is printed?**

Answer:

total in class Final 1 is obtained by adding the instance variable with instance variable (a) of the superclass that are accessed by using *super* keyword and 80

**Question 3**

Answer: the method in line 13 will override the method in line 6

**Question 4**

Answer: this program will produce error

Reason: **Class B item1=new Class A ();**

1. **Method1 in class A consists parameter of (int a). Method1 require an argument. Method overloading**

**Corecct Syntax**

**ClassA item1=new ClassA();**

**Item1.method(5)**

**Question 5**

Answer: method1 in class C will be executed due to this declaration

ClassA item1=new ClassC(); object of class C is created. Therefore method1 in class c is execured

**Question 6**

**Error line 27**: **Circle(radius);**

**Correction**: **super(radius);**

**Explanation**: to access radius data member from class Circle, *super* keyword must be used

**Error line 33**: **return getArea()\*length;**

**Correction:** **return super.getArea()\*length**

**Explanation**: the getArea() method exist in class Circle. To use it, *super* keyword must be used

**Question 7**

**Overriding:** showValue (int ar) and showValue(int a)

**Overloading:** showValue(int a), showValue(String a), showValue(double a)

**Question 8**

1. Output of program 8.26



1. Output of program 8.27

A picture containing object, clock

Description automatically generated

ii)

**method overloading**: Program 8.27

**method overriding:** Program 8.26

iii)

|  |  |
| --- | --- |
| Method overloading | Method overriding |
| Methods are having the same name with different parameters | Methods have same name and same signature |
| May or may not require inheritance | Always require inheritance |