01)

```
class A{
	//---------super class----------

}class B extends A{
	//--------- sub class of  A ----------

}class C extends A{
	//---------sub class of B----------

}class D extends C{
	//---------sub class of D----------

}class E extends C{
	//---------sub class of E----------

}

class Example{
	public static void mian(String args[]){

		}
}
```

02)

- Inheritance in Object-Oriented Programming allows a new class (child or subclass) to inherit attributes and methods from an existing class (parent or superclass). This promotes code reusability and logical structuring.

Real-World Examples:

Animals:

Parent Class: Animal
Attributes: name, age
Methods: eat(), sleep()
Child Classes: Dog, Cat
Dog: Inherits name, age, eat(), sleep(); Adds breed, bark()
Cat: Inherits name, age, eat(), sleep(); Adds color, meow()

Vehicles:

Parent Class: Vehicle
Attributes: make, model, year
Methods: start(), stop()

Child Classes: Car, Bicycle
Car: Inherits make, model, year, start(), stop(); Adds number_of_doors, honk()
Bicycle: Inherits make, model, year, start(), stop(); Adds type_of_bike, ring_bell()

03)

I. Code Reusability:

- Child classes reuse attributes and methods from the parent class, avoiding code duplication.
- Common functionalities are implemented once in the parent class and used by all child classes.

II. Time Savings:

- Development is faster as developers focus on new features specific to child classes without re-implementing shared functionalities.
- Simplified maintenance as changes in the parent class propagate to all child classes.

III. Error Prevention:

- Ensures consistency by providing a uniform implementation of shared methods and attributes.
- Centralized bug fixes in the parent class benefit all child classes.
- Promotes encapsulation, reducing the risk of accidental interference.

04)

I.

```
class Student{
	//---------super class----------

}class GraduateStudent extends Student{
	//---------sub class of Student----------

}class UndergraduateStudent extends Student{
	//---------sub class of Student----------

}
class Example{
	public static void mian(String args[]){

		}
}
```

II.

```
class Shape{
            //---------super class----------

}class Circle extends Shape{
            //---------sub class of Shape----------

}class Triangle extends Shape{
            //---------sub class of Shape----------

}class Rectangle extends Shape{
            //---------sub class of Shape ----------

}class Sphere extends Shape{
            //---------sub class of Shape----------

}
class Cube extends Shape{
            //---------sub class of Shape----------

}

class Example{
            public static void mian(String args[]){

                        }
}
```

III.

```
class Loan{
            //---------super class----------

}class CarLoan extends Loan{
            //---------sub class Loan----------

}class HomeImprovementLoan extends Loan{
            //---------sub class of Loan----------

}class MortgageLoan extends Loan{
            //---------sub class of Loan----------

}
```

```
class Example{
            public static void mian(String args[]){

                        }
}
```

IV.

```
class Employee{
            //---------super class----------

}class Employee extends Employee{
            //---------sub class Employee----------

}class Faculty extends Employee{
            //---------sub class Employee----------

}class Staff extends Employee{
            //---------sub class Employee----------

}
class Example{
            public static void mian(String args[]){

                        }
}
```

V.

```
class BankAccount{
            //---------super class----------

}class CheckingAccount extends BankAccount{
            //---------sub class of BankAccount----------

}class SavingsAccount extends BankAccount{
            //---------sub class of Bank Account----------

}
class Example{
            public static void mian(String args[]){

                        }
}
```

05)

```
class Shape{
	//---------super class----------

}class TwoDimensionalShape extends Shape{
	//---------sub class of Shape----------

}class ThreeDimensionalShape extends Shape{
	//---------sub class of Shape----------

}class Circle extends TwoDimensionalShape{
	//---------sub class of TwoDimensionalShape ----------

}class Square extends TwoDimensionalShape{
	//---------sub class of TwoDimensionalShape ----------

}
class Triangle extends TwoDimensionalShape{
	//---------sub class of TwoDimensionalShape ----------

}
class Sphere extends ThreeDimensionalShape{
	//---------sub class of ThreeDimensionalShape ----------

}class Cube extends ThreeDimensionalShape{
	//---------sub class of ThreeDimensionalShape ----------

}
class Tetrahedron extends ThreeDimensionalShape{
	//---------sub class of ThreeDimensionalShape ----------

}

class Example{
	public static void mian(String args[]){

		}
}
```

06)

07)

08)

09)

A. True
B. True
C. True
D. True
E. True
F. True

10)

- Line 2: printA(); This will cause an error because printA() is a method of class A, and there is no inheritance or object reference to call this method from class B.

- Line 4: System.out.println("a : "+a); This line will also cause an error because the variable a is not defined in class B. It is defined in class A, and there's no direct access to it from class B.

11)

A. False
B. True
C. True
D. False

12)

13) A

14) The program will not compile because class B does not provide a constructor to call the superclass A's constructor, which requires an integer argument.

15)

- Line 9: B b2=a1; - This will cause an error because a1 is an instance of A, and you cannot assign an object of a superclass to a variable of a subclass without an explicit cast.

- Line 10: b2=c1; - Similar to line 9, this will cause an error because c1 is an instance of C, which is not a subclass of B.

- Line 12: C c2=a1; - This will cause an error because a1 is an instance of A, and you cannot assign an object of a superclass to a variable of a subclass without an explicit cast.

- Line 13: c2=b1; - This will cause an error because b1 is an instance of B, which is not a subclass of C.

- Line 15: D d2=a1; - This will cause an error because a1 is an instance of A, and you cannot assign an object of a superclass to a variable of a subclass without an explicit cast.

- Line 16: d2=b1; - This will not cause an error because b1 is an instance of B, and D is a subclass of B.

- Line 17: d2=c1; - This will cause an error because c1 is an instance of C, which has no relationship with D.

16)

B. A method with the same name completely replaces the functionality of a method earlier in the hierarchy.

17) D. class AnimalShape{}

18)

19)

20)

- super(i); - This will call the Super(int i) constructor with the value of i passed to the Sub constructor.
- super(100); - This will call the Super(int i) constructor with a literal value of 100.

21)

In Java, you invoke a superclass constructor from a subclass using the super keyword followed by parentheses containing any arguments needed by the superclass constructor. This call to super must be the first statement in the subclass constructor.

22)

In Java, if a subclass defines an attribute (field) with the same name as an attribute in its superclass, the subclass's attribute hides the superclass's attribute. This is known as field hiding, not overriding. Overriding is a concept that applies to methods, not fields.

23)

24) F. hi hi, followed by an exception

25)

26)

Runtime polymorphism allows a subclass to override a method in its superclass. In this example, Figure's area() method is overridden by Rectangle and Triangle. When a superclass reference (Figure figref) points to a subclass object (Rectangle or Triangle), the subclass's overridden area() method is called at runtime.

27)

28)

29)

30)

A. True
B. True
C. True
D. False

31)

B. Instance variables declared in this class or any superclass
D. Instance methods declared in this class or any superclass.
E. The keyword "this".

32)

33)

34)

35) A. ABC

36) ------------------True---------------------

B. Has-a relationships always rely on instance variables.

C. Has-a relationships always require at least two class types.

37) D. Prints Sub : null -> Panadura