IS-A vs HAS-A Relationship

- Is-A and Has-A Relationship shows how classes or objects are related to each in a Object oriented design.
- Is-A relationship is based on inheritance and interface implementation. ‘IS-A’ is way of saying “this thing is a type of that thing” .It shows whether a class should be inherited by other class or not.

House IS-A building, dog IS-A animal etc.

- HAS-A relationship is based on class rather than inheritance. It shows whether a class should contain object of other class or not.

e.g. Building HAS-A bathroom

Polymorphism

- Polymorphism is basically the ability of an object to take many form. ‘Any object which passes more than one IS-A test is considered as polymorphic’.
- The java the concept of polymorphism is supported in terms of :

✓ Method overriding

✓ Method overloading

✓ Polymorphic references

✓ Polymorphic arguments and parameters

Polymorphic references and Array

- **The most common use of polymorphism is that reference of superclass can be used to refer objects of subclass.**
- **In polymorphic array reference of superclass can refer any object of subclasses:**
- **public class Test {**
- **public static void main(String[] args)**
- {
- Animal[] a = **new Animal[4];**
- a[0] = **new Cat();**
- a[1] = **new Dog();**
- a[2]= **new Duck();**
- a[3] = **new Snake();**
- **for(int i=0; i<a.length; i++)**
- {
- a[i].sound();}

Polymorphic arguments and Parameters

- **Polymorphic Parameter:**

Polymorphic parameters are nothing but the parameter of type "superclass".

e.g. If we have a method which contain polymorphic parameters then in arguments we can pass objects of any subclass type.

- **Polymorphic Arguments:**

Polymorphic arguments are the arguments of type "subclass".

- **public void go(Animal a)**
- {
- a.sound();
- } // in class Animal
- **public static void main(String[] args)**
- {
- Animal a = **new Animal();**
- Cat c = **new Cat();**
- Dog d = **new Dog();**
- a.go(d);
- }

Data Abstraction and Abstract Class

- Abstraction is a design concept on which we only declare functionality but doesn't define it because we don't know about them at design point.
- Abstract Keyword :

This is a special keyword which is used as a non-access modifiers with classes and methods.

- Abstract Keyword with class:

If ‘abstract’ keyword is used with a class, then no one can instantiate that class and these classes are known as ‘abstract classes’.

- Abstract Keyword with method:

If ‘abstract’ keyword is used with method, then it must be overridden in first concrete class????

Abstract Class Vs Concrete Class

- The classes which cannot be instantiated are known as Abstract class.
- Concrete class are those classes which can be intantiated

Abstract Methods and it's properties

- The abstract Method is that method which doesn't contains any body and must be overridden in first concrete class.
- Properties:

✓ Abstract class should always use the keyword 'abstract' and in declaration it must be without body.

✓ The class of abstract method must be an abstract class. Abstract method cannot be declare in a concrete class.

✓ The abstract method must be overridden by first concrete class which extends the abstract class.

Class-Object

- The class Object is mother of all classes in java.
- Any class which doesn't extends any class extends class Object.
- Hence every class in java which extends other class is polymorphic, i.e. every class in java "IS-A type of Object".

e.g.

Public class Test{}

Means

Public class Test extends Object{}

Important Method of Class - Object

- equals() method:

This method checks the equality of two objects, whether they are equal or not

Animal a = new Cat();

Animal b = new Dog(); → a.equals(b);

- getClass() method:

This method returns the real class name of it's corresponding object.

e.g. b.getClass();

- hashCode()

This method finds the hashCode of a object

e.g. b.hashCode();

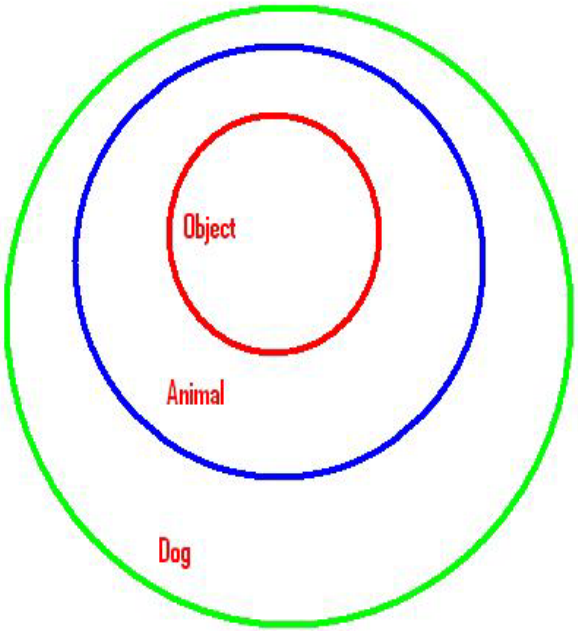
- toString():

This method returns the string representation of an object

Class of reference variable

- During the call of method of an object, the compiler searches the **method in class of reference variable** and not in the actual class of the object.
- **public static void main(String[] args)**
- {
- Object o = **new Animal();**
- o.eat();
- }

Object creation in Java-heap



Instance of operator and Object Casting

- Instanceof Operator:

Instanceof Operator is used to check whether the object is of type of a given class or not. If yes, it returns true otherwise false.

Animal a = **new Cat();**

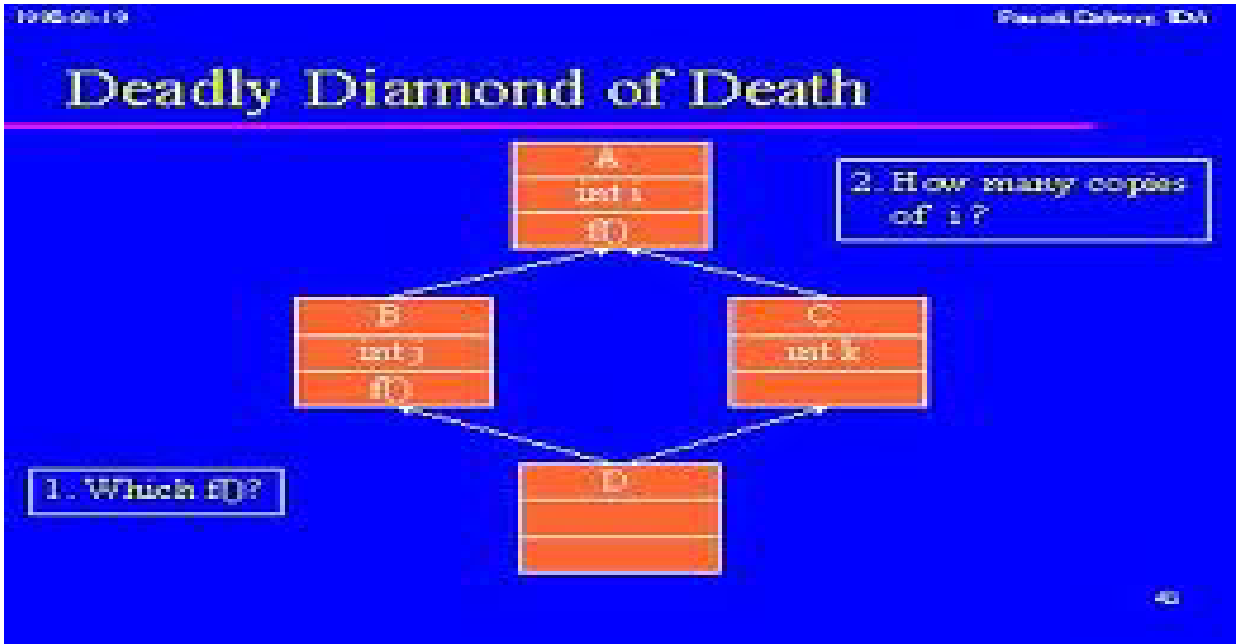
Object o = a;

System.*out.println(a* ***instanceof Animal);***

- *Any object can be casted to it's original class type by object casting.*
- **public static void main(String[] args)**
- {
- Animal a = **new Cat();**
- Object o = a;
- Cat c = (Cat)o;
- c.sound();
- }

Limitation of Multiple inheritance

- Multiple Inheritance is the mechanism of extending more than one superclass.
- Java doesn't supports multiple inheritance due to it's limitation in "deadly diamond of death"



Solution of DDoD in java - Interfaces

- In java the alternative of multiple inheritance is INTERFACES
- INTERFACES are nothing but the pure abstract class which can be implement by any other class along with extending other classes