**Advanced Programming Tutorial & Lab - Java Basics**

Objective: The objective of this tutorial is to refresh abstract classes and polymorphism.

# Review Questions:

* 1. What does the term polymorphism refer to?
  2. What is dynamic binding and how does it work?
  3. How can typecasting be used with object types?
  4. What is the instanceof operator used for?
  5. What does visibility refer to and how does it affect inheritance/accessibility between classes?
  6. What are abstract classes and when should you use them?
  7. What restrictions are placed on subclasses of an abstract class?
  8. How do you implement an abstract class – is it any different from a “normal” class?
  9. What is an interface in java and when should you use one?
  10. How is an interface different from an abstract class?
  11. What relationship is there between abstract classes/interfaces and polymorphism?

1. An estate agency has two schemes for selling houses, sale by auction and sale by negotiation. For sale by negotiation the charge is based on sale price \* commission rate while for auctioned houses the charge is: (actual price – base price) \* bonus rate + 2000.

A program is required to compute charges (polymorphically) and print. You are required to modify the program which only prints the address.

Sale by Negotiation

|  |  |  |
| --- | --- | --- |
| Address | Actual price | Commission rate |
| 34 Kew Crt | 340,000 | 0.05 |
| 2 John St | 420,000 | 0.03 |

Sale by Auction

|  |  |  |  |
| --- | --- | --- | --- |
| Address | Base price | Actual price | Bonus rate |
| 5 Bet St | 560000 | 565000 | 0.10 |
| 12 Ron Dr | 240,000 | 290,000 | 0.15 |

Steps Needed:

* 1. In the superclass Sale add a method named double computeCharge()
  2. As it cannot be defined (no common scheme) make this method abstract
  3. This requires the class also to be abstract
  4. Override this method in both subclasses using the scheme specified
  5. Now the method computeCharge() can be called polymorphically in the class ManageSales

class Sale

{ private String address; public Sale(String add)

{ address = add; } public String getAddress()

{ return address; }

}

class NegotiatedSale extends Sale

{

private String address; private double commRate; private double salePrice;

public NegotiatedSale(String add,double price,

double cRate )

{ super(add); salePrice = price; commRate = cRate;

}

}

**class** AuctionSale **extends** Sale

{ **private double** basePrice; **private double** actualPrice; **private double** bonusRate;

**public** AuctionSale(String add, **double** bPrice, **double** aPrice,

**double** bRate )

{ super(add); basePrice = bPrice; actualPrice = aPrice; bonusRate = bRate;

}

}

**public class** ManageSales

{ **public static void** main(String args[])

{ Sale s[] = **new** Sale[4];

s[0] = **new** NegotiatedSale("34 Kew crt",340000,0.05); s[1] = **new** AuctionSale("5 Bet crt",560000, 565000, 0.10);

s[2] = **new** AuctionSale("12 Ron dr",240000, 290000, 0.15); s[3] = **new** NegotiatedSale("2 John st",420000,0.03);

**for** (**int** i=0; i<4; i++) System.out.println(s[i].getAddress());

}

}

## Lab Exercise

**Implement the solution for question 2**