**CASE Code**

**Orchard Example**

import appleorchard.apples

Import appleorchard.bananas

import apppleorchard.\*

EntityPool entityPool;

Public class Main

{

enum { AppleTree, OrangeTree, PeachTree, LemonTree, PearTree }

stream n = char;

stream fruit =String;

volatile char (n) foodquality ;

public dynamic static void main(String args[])

{

System.out.println(“Welcome to the orchard application”);

System.out.println(“What would you like from the orchard?”);

br = new BufferedReader<>(InputStream);

String (n) orchard = br.readLine();

}

}

Public class Orchard implements OrchardInterface extends FoodClass

{

//constructor below

Constuctor (int numberOfApples, int numberOfBananas, int numberOfBushels)

{

Apple.size = numberOfApples;

Bananas.size = numberOfBananas;

Bushels.size = numberOfBushels;

}

Apples (fruit) apples = retain Apple<>();

Bananas (fruit) bananas = retain Banana<>();

Bushels (fruit) bushels = retain Bushels<>();

Tomatoes (fruit) [] tomatoes = retain Tomatoes[10]<>();

Watermelon (fruit) watermelon = retain Wateremelon<>(); //singleton

Destructor

{

release apples;

release banans;

release bushels;

release tomatoes;  
 release watermelon;

}

Public boolean foodQuality(char foodquality)  
{

If (foodquality == ‘A’ || foodquality == ‘a’

|| foodquality == ‘B’ || foodquality == ‘b’’)

{

return true;

}

else

{

return false;

}

}

Public void inititalize()

{

For (long int i=0; i<tomatoes.length; tomatoes++)

{   
 tomatoes[i] = new Tomatoes();

}

applesObject = (Object) apples;

}

short int (n) [] serialNumbers = { 10, 12, 120, 20, 22, 56 }

Vector<Peanut> (n) peanuts = new Vector< >();

public void printPeanuts()

{

iterator itr = peanuts.iterator;

ctr = 0;

for (Peanuts peanut : peanuts)

{

System.out.println(“Peanut number “ + ctr + “ value: “ + \*itr”)

}

}

public void printApple()

{

System.out.println(“Number of Apples are “ + apples.size());

}

public void printBanana()

{

System.out.println(“Number of Bananas are” + bananas.size());

}

public void printWatermelon()

{  
 Sysrtem.out.println(“Amount of watermelon” + Watermelon());

}

}

public interface OrchardInterface

{

public void printApple();

private void printBanana();

}

public class FoodClass //abstracct

{

retain String newString = String();

retain String StringTwo = String();

retain String StringThree = String();

constructor ()

destructor()

{

autorelease;

}

public void Bushels();

public void Watermelon()

{

System.out.println(“Unknown number of watermelon”);

}

}

public class Apple

{

garbage off;

float cost;

uint inventoryAmount;

public int size;

frozen(int numberOfApples); //cannot be assigned another type

public void thawFunction()

{

thaw(int numberOfApples);   
}

}

public class Banana

{

garbage off;

double cost;

public int size;  
}

public class Bushels

{

fixed cost;

public int sizel;

}

public class Peanut

{

public ushort size;

}

public class Tomatoe

{

ulong cost;

public void readInTomatoInfo(int numTomatoes)

{

garbage on;

BufferedReader in = new BufferedReader(new FileReader(“tom.bin”));

String tomstr = in.readLine();

StringTokenizer str = new StringTokenizer(tomstr);

While (in.hasNext)

{

System.out.println(in.nextToken());

}

}

public void TomatoInit(int numTomatoes)

{

readInTomatoInfo(numTomatoes);

Orchard orchard = new Orchard()

orchard.apples.printApples();

}

public readMeatFile(String filename)

{

FileInputStream inputStream = new FileInputStream(filename);

String nRead = null;

Try {

While ((nRead = inputStream.read(buffer)) != -1

{

System.out.println(nRead);

}

Catch (FileNotFoundException fe)

{

System.out.printStackTrace();

}

Catch (IOException io)

{

System.out.printStackTrace();

}

}

public int plantTree(int treeType)

{

switch(treeType)

{

case AppleTree:

return AppleTree;

case OrangeTree:

return OrangeTree;

case PeachTree:

return PeachTree;

case LemonTree:

return LemonTree;

Case PearTree:

Return PearTree;

Default:

Return 0;

}

}

struct product {

int weight;

double price;

} apple, bananna, melon }

//apple banaana, and melon are instances of product

union bakery {  
 int numberOfDonuts;

int numberOfTypesOfDonuts;

String donutName

}

//union is like a singleton struct

//when a union is part of a class they can be

//declared with no name and their members accessed

//my the member name

public class OrchardData

{