BITS Pilani
Pilani Campus

Object Oriented Programming
CS F213

J. Jennifer Ranjani
email: jennifer.ranjani@pilani.bits-pilani.ac.in
Chamber: 6121 B, NAB
Consultation: Appointment by e-mail

Overriding, Abstract Class and Arrays

BITS Pilani
Pilani Campus

BITS Pilani
Pilani Campus

Method Overriding

What is Overriding?

- In a class hierarchy, when a method in a subclass has the same name and type signature as a method in its superclass, then the method in the subclass is said to override the method in the superclass.
- When an overridden method is called from within a subclass, it will always refer to the version of that method defined by the subclass.
- The version of the method defined by the superclass will be hidden.
- A subclass may call an overridden superclass method by prefixing its name with the ‘super’ keyword and a dot (.).

Overriding - Example

```
class CheckingAccount extends BankAccount
{
private static final float TRANS_FEE = 25;
private static final int FREE_TRANS = 2;
private float TransCount =0;

CheckingAccount(int acc,String name,float amt) {
super(acc,name,amt);   }

void deductFee()   {
if(TransCount > FREE_TRANS)
 {
float fee = (TransCount-
     FREE_TRANS)*TRANS_FEE;
super.withdraw(fee);
TransCount=0;}  }
void deposit(float amount)
{
TransCount++;
super.deposit(amount);
}
void withdraw(float amount)
{
TransCount++;
super.withdraw(amount);
}

}
```

Overriding - Example

```
class TestAccount{
public static void main(String[] args) {

CheckingAccount ca= new CheckingAccount(111,"Ankit",5000);

System.out.println("Initial: "+ca.getBalance());

ca.deposit(1000);
ca.withdraw(2000);
ca.deposit(6000);
System.out.println("After three Transactions: " + ca.getBalance());

ca.deductFee();

System.out.println("After fee Deduction: " + ca.getBalance());
}}
```

innovate achieve lead

BITS Pilani
Pilani Campus

‘Final’ Keyword

Java Final Keyword

- Makes variable a constant
- Prevents Method Overriding
- Prevents Inheritance

Blank or uninitialized final variable

- A final variable that is not initialized at the time of declaration is known as blank final variable.

- It can be used when variable is initialized at the time of object creation and should not be changed after that.
  - Eg. Pan card

- It can be initialized only once (preferably within a constructor).

Final blank variable

Example 1:

```
class first{

 public static void main(String
     args[]){
    final int i;
    i=10;

      System.out.println("s1: "+i);
      i=20;  // Error

   }
}
```

Example 2:

```
class first{
final int i;
i=10 // Error
first(){
i=10;
}

 public static void main(String
     args[]){

       System.out.println("s1: "+new
     first().i);
    }
}
```

Static Blank Final Variable

- A static final variable that is not initialized at the time of declaration is known as static blank final variable. It can be initialized only in static block.

```
class A{
  static final int data;//static blank final variable
  static{ data=50;}
  public static void main(String args[]){
    System.out.println(A.data);
 }
}
```

Questions?

- Is final method inherited?
    - YES. But it cannot be overridden

- Can we declare a constructor final?
    - NO. Constructor is not inherited

innovate achieve lead

BITS Pilani
Pilani Campus

Run Time Polymorphism

Dynamic Method Dispatch

- Method overriding is one of the ways in which Java supports Runtime Polymorphism.
- Dynamic method dispatch is the mechanism by which a call to an overridden method is resolved at run time, rather than compile time.
- An overridden method is called through the reference variable of a superclass.
- The determination of the method to be called is based on the object being referred to by the reference variable.
- **Upcasting**: The reference variable of Parent class refers to the object of Child class.

Bank - Example

```
class TestAccount{
public static void main(String[] args) {

Scanner sr = new Scanner(System.in);

System.out.println("Enter 1 for new customers (< 1 year) and 0 for others");
int yr = sr.nextInt();

BankAccount ba;

if (yr==1)
ba = new BankAccount(111,"Ankit",5000);
else
ba = new CheckingAccount(111,"Ankit",5000);
```

Bank - Example

```
System.out.println("Initial: "+ba.getBalance());

ba.deposit(1000);
ba.withdraw(2000);
ba.deposit(6000);
System.out.println("After three Transactions: " + ba.getBalance());

ba.deductFee();        //ERROR

System.out.println("After fee Deduction: " + ba.getBalance());
sr.close();

}}
```

Solution 1

- Create an empty method in the Bank Account class

```
void deductFee()
{
}
```

- Meaningless, Isn’t it?

Solution 2 – Abstract Class

```
abstract class BankAccount{
private int acc;
private String name;
private float amount;

BankAccount(int acc,String name,float amt)
{
this.acc = acc;
this.name = name;
this.amount = amt;   }

void setAcc(int acc)   {
this.acc = acc;    }

void setName(String name)  {
this.name = name;   }

float getBalance(){
return amount;}

void deposit(float amount)   {
this.amount = this.amount+amount; }

void withdraw(float amount)  {
if (this.amount < amount)
System.out.println("Insufficient
    Funds. Withdrawal Failed");
else
this.amount=this.amount-amount;  }

abstract void deductFee();
}
```

Static vs. Dynamic Binding (Early vs. Late Binding)

- Static binding happens at compile-time while dynamic binding happens at runtime.
- Binding of private, static and final methods always happen at compile time since these methods cannot be overridden.
- When the method overriding is actually happening and the reference of parent type is assigned to the object of child class type then such binding is resolved during runtime.
- The binding of overloaded methods is static and the binding of overridden methods is dynamic.

Arrays

- Syntax to declare an array
  - int[] arr;
  - int []arr;
  - int arr[];

- Instantiation of an array
  - arr = new int[size];

- Arrays can be accessed using
  - Simple for loop
  - For each loop
  - Labelled for loop

For each loop

```
int arr[]={12,23,44,56,78};
    //Printing array using for-each loop
    for(int i:arr){
        System.out.println(i);
    }
```

Labelled For Loop

```
aa:
    for(int i=1;i<=3;i++){
        bb:
            for(int j=1;j<=3;j++){
                if(i==2&&j==2){
                    break bb;
                }
                System.out.println(i+" "+j);
            }
    }
```

Array Index Out of Bounds

- Array indices always start with `0`, and always end with the integer that is one less than the size of the array
  - The most common programming error made when using arrays is attempting to use a nonexistent array index

- When an index expression evaluates to some value other than those allowed by the array declaration, the index is said to be *out of bounds*
  - An out of bounds index will cause a program to terminate with a run-time error message
  - Array indices get out of bounds most commonly at the *first* or *last* iteration of a loop that processes the array: Be sure to test for this!

Array of Characters is not a String!!!

- An array of characters is conceptually a list of characters, and so is conceptually like a string
- However, an array of characters is not an object of the class `String`
  - `char[] a = {'A', 'B', 'C'};`
  - `String s = a; //Illegal!`
- An array of characters can be converted to an object of type `String`, however
  - `char[] a = {'A', 'B', 'C'};`
  - `String s = new String(a);`
  - `System.out.println(s);`
  - `s = new String(a,1,2);`
  - `System.out.println(s);`

Copying a Java Array

public static void arraycopy(Object src, **int** srcPos,Object dest, **int** destPos, **int** length)

- arraycopy method of the System class is used to copy an array to another.

```
int a[]= {2,3,5};
 int b[] = new int[a.length];

 System.arraycopy(a, 1, b, 0, a.length-1);

 for(int i=0;i<b.length;i++)
 System.out.print(" "+b[i]);
```

Output:

```
3 5 0
```

Array Class

static type	**binarySearch(type[] a, type key)** **Searches the specified array of type for the specified value using the binary search algorithm.**
static boolean	equals(type[] a, type[] a2) Returns true if the two specified arrays of type are equal to one another.
static void	fill(type[] a, type val) Assigns the specified type value to each element of the specified array of type.
static void	fill(type[] a, int fromIndex, int toIndex, type val) Assigns the specified type value to each element of the specified range of the specified array of types.
static void	sort(type[] a) Sorts the specified array of type into ascending numerical order.
static void	sort(type[] a, int fromIndex, int toIndex) Sorts the specified range of the specified array of type into ascending numerical order.
	type = byte, char, double, float, int, long, short, Object

Array Class - Example

```
int a[]= {2,3,5,1,4,7};

for(int i=0;i<a.length;i++)
System.out.print(a[i]+" ");

System.out.println();
Arrays.sort(a,0,4);
System.out.println(Arrays.toString(a));

Arrays.sort(a);
System.out.println(Arrays.toString(a));

System.out.println("Binary Search for 5 is "+Arrays.binarySearch(a, 5));
```

Output:

```
2 3 5 1 4 7
[1, 2, 3, 5, 4, 7]
[1, 2, 3, 4, 5, 7]
Binary Search for 5 is 4
```

Array Class - Example

```
int a[]= {2,3,5,1,4,7};

System.out.println(Arrays.toString(Arrays.copyOf(a, a.length)));

System.out.println(Arrays.toString(Arrays.copyOfRange(a, 1,4)));

Arrays.fill(a,4,a.length,1);
System.out.println(Arrays.toString(a));

Arrays.fill(a,1);
System.out.println(Arrays.toString(a));
```

Output:

```
[1, 2, 3, 4, 5, 7]
[2, 3, 4]
[1, 2, 3, 4, 1, 1]
[1, 1, 1, 1, 1, 1]
```