Mahindra École Centrale
COLLEGE OF ENGINEERING

Object Oriented Programming JAVA

Dr. Prafulla Kalapatapu
Computer Science Engineering
Mahindra Ecole Centrale
prafulla.kalapatapu@mechyd.ac.in

Mahindra École Centrale
COLLEGE OF ENGINEERING

Polymorphism

Method overriding

Mahindra École Centrale
COLLEGE OF ENGINEERING

- What is a method overriding

  Redefining super class method in the sub class with the following rules :
  All rules has to satisfy.

  1. Return type should be same
  2. Method signature should be same
  3. subclass method should not have weaker privilege than super class method (greater or equal is ok)

Program on method overriding

Mahindra École Centrale
COLLEGE OF ENGINEERING

```
class A {
void m1() {
        System.out.println("super m1()"); }
}
class  B extends A {
public void m1() {
        System.out.println("sub m1()"); }
}
class Sample {
public static void main(String …a) {
        B b1=new B();
        b1.m1();
}
}
```

Overriding method

Overridden method

O/P : sub m1()

- Method resolution always taken care by JVM based on runtime object.
- Overriding is also considered as runtime polymorphism, dynamic polymorphism, late binding.
- What is the pre-requisite for method overriding.

  inheritance

Method hiding

Mahindra École Centrale
COLLEGE OF ENGINEERING

- Static methods cant be overridden.

- We can do method hiding, instead of overriding. That means both super and subclass methods should be static.

Ex:
```
class A {
public static void m1() { ……}
}
class B extends A {
public static void m1() {…….}
}
```

In method hiding, method resolution will always taken care by compiler based on reference type.

Overloading vs Overriding

Property	Overloading	Overriding
Method names	Must be same	Must be same
Arguments	Number of, type of, order of arguments either of them differ	Should be same
Method signature	Must be different	Must be same
Return type	No restriction	Must be same
Access specifier	No restriction	Weaker privilege is not allowed in sub class
private, final, static methods	Can be overloaded	Cannot be override
Method resolution	Always taken care by compiler based on reference type	Always taken care by JVM based on runtime object
Also known as	Static polymorphism, compile time polymorphism, early binding	Dynamic polymorphism, runtime polymorphism, late binding

Overriding Vs Method hiding

- All rules of method hiding and overriding are exactly same expect the following differences.

Method overriding	Method hiding
Both super and sub class method's should be non-static	Both super and sub class method's should be static
Method resolution taken care by JVM based on runtime object	Method resolution taken care by compiler based on reference type.
It is considered as runtime polymorphism, dynamic polymorphism, late binding	It is considered as compile time polymorphism, static polymorphism, early binding

Mahindra
École Centrale
COLLEGE OF ENGINEERING

String, String Buffer, String Builder

1. String

Mahindra École Centrale
COLLEGE OF ENGINEERING

- String is a final class.
- String is immutable(can not change contents) class.

Ex 1:

```
class Sample{
public static void main(String... a){
String s1="Hello";
System.out.println(s1.hashCode());
System.out.println(s1.concat("students"));
System.out.println(s1.concat("students").hashCode());
}
}
```

//hashCode() method returns unique integer value for each object.

hashCode() from String class

- Syntax: public int hashCode()

- The hash code for a String object is computed as:

```
public  int hashCode()
{
return  s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1];
}
```

s[0] is characters from string
n is length of the string

equals() from String class

Mahindra École Centrale
COLLEGE OF ENGINEERING

- Syntax:

  public boolean equals(Object o2)

- Calling:

  o1.equals(o2);

- Code:

public boolean equals(Object o2)
{
If length's are equal
It checks character by character.
}

If two objects o1 and o2 contents are equal , it returns true ,otherwise false

Ex:

"hello".equals("hello"); //returns true

Immutable classes

Mahindra École Centrale
COLLEGE OF ENGINEERING

- Immutable means: contents can't be modified
- Ex:

  String class
- Are there any other classes whose objects are immutable?

  Yes. All Wrapper classes(Integer,Byte,Character,Float,double) are immutable

2. StringBuffer

Mahindra École Centrale
COLLEGE OF ENGINEERING

- StringBuffer is a final class.
- StringBuffer is mutable(can change contents) class.
- StringBuffer is threadsafe or synchronization

Constructors

Mahindra École Centrale
COLLEGE OF ENGINEERING

StringBuffer() Constructs a string buffer with no characters in it and an initial capacity of 16 characters.

StringBuffer(CharSequence seq) Constructs a string buffer that contains the same characters as the specified CharSequence.

StringBuffer(int capacity) Constructs a string buffer with no characters in it and the specified initial capacity.

StringBuffer(String str) Constructs a string buffer initialized to the contents of the specified string.

append() from StringBuffer

Mahindra École Centrale
COLLEGE OF ENGINEERING

```
class Sample{
public static void main(String... a){
StringBuffer  sb=new StringBuffer("Hello");
System.out.println(sb);
sb.append(“Students");
System.out.println(sb);
}
}

//first S.o.p statement prints Hello
//first S.o.p  it prints HelloStudents
```

capacity() from StringBuffer

Mahindra École Centrale
COLLEGE OF ENGINEERING

```
class Sample
{
public static void main(String... a)
{
StringBuffer  sb=new StringBuffer();
System.out.println(sb.capacity());
}
}
```

//it prints 16

capacity() from StringBuffer

Mahindra École Centrale
COLLEGE OF ENGINEERING

```
class Sample{
public static void main(String... a){
StringBuffer  sb=new StringBuffer();
System.out.println(sb.capacity());
StringBuffer  sb1=new StringBuffer(50);
System.out.println(sb1.capacity());
StringBuffer  sb2=new StringBuffer("bits");
System.out.println(sb2.capacity());
}
}

//it prints 16
//it prints 50
//it prints 16+4=20
```

3. StringBuilder

Mahindra École Centrale
COLLEGE OF ENGINEERING

- StringBuilder is a final class.
- StringBuilder is mutable(can change contents) class.
- StringBuilder is not threadsafe(not synchronized).

constructor

Mahindra École Centrale
COLLEGE OF ENGINEERING

Constructor	Description
StringBuilder()	Creates an empty string builder with a capacity of 16 (16 empty elements).
StringBuilder(CharSequence cs)	Constructs a string builder containing the same characters as the specified CharSequence, plus an extra 16 empty elements trailing the CharSequence.
StringBuilder(int initCapacity)	Creates an empty string builder with the specified initial capacity.
StringBuilder(String s)	Creates a string builder whose value is initialized by the specified string, plus an extra 16 empty elements trailing the string.

StringBuilder

Mahindra École Centrale
COLLEGE OF ENGINEERING

```
class Sample{
public static void main(String... a){
StringBuilder  sb=new StringBuilder();
System.out.println(sb.capacity());
StringBuilder  sb1=new StringBuilder(50);
System.out.println(sb1.capacity());
StringBuilder  sb2=new StringBuilder("bits");
System.out.println(sb2.capacity());
}
}

//it prints 16
//it prints 50
//it prints 16+4
```

StringBuffer vs StringBuilder

What is the difference between StringBuffer and StringBuilder?

- StringBuffer class is synchronized and StringBuilder is not.
- when programmer wants to use several threads, he should use StringBuffer as it gives reliable results.
- if only one thread is used , StringBuilder is preferred , as it is improves execution time.

StringBuffer vs StringBuilder

Note: equals method from Object class.

```
public boolean equals(Object  o2)
{
return   this==o2;
}
```

In StringBuffer and StringBuilder:

hashCode() and equals() methods inherited from Object class.

In String class:

hashCode() and equals() methods overridden in String class

StringBuilder

Mahindra École Centrale
COLLEGE OF ENGINEERING

```
class Sample{
public static void main(String... a){
StringBuilder  sb1=new StringBuilder("hello");
StringBuilder  sb2=new StringBuilder("hello");
System.out.println(sb1.equals(sb2)); //it checks two references/hashcode are
    equal   or not( not the contents)
System.out.println(sb1.hashCode());
System.out.println(sb2.hashCode());
}
}
```

StringBuffer

Mahindra École Centrale
COLLEGE OF ENGINEERING

```
class Sample{
public static void main(String... a){
StringBuffer sb1=new  StringBuffer("hello");
StringBuffer  sb2=new StringBuffer("hello");
System.out.println(sb1.equals(sb2)); //it checks two hashcode are equal   or not(
    not the contents)
System.out.println(sb1.hashCode());
System.out.println(sb2.hashCode());
}
}
```

String

Mahindra École Centrale
COLLEGE OF ENGINEERING

```
class Sample{
public static void main(String... a){
String  sb1=new String("hello");
String  sb2=new String("hello");
System.out.println(sb1.equals(sb2)); //it checks contents
System.out.println(sb1.hashCode());
System.out.println(sb2.hashCode());
}
}
```