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

String, String Buffer, String Builder Examples

Capacity()

Mahindra École Centrale
COLLEGE OF ENGINEERING

int capacity()
{
if we create StringBuffer object (with no argument or initial capacity)
{
StringBuffer s1=new StringBuffer();
capacity=16;
or
StringBuffer s2=new StringBuffer(30);
capacity=30;
}
if we create StringBuffer object (with string object as an argument).
{
StringBuffer s3=new StringBuffer("hello");
capacity=16+length of the given string;
=16+5=21
}

Mahindra École Centrale
COLLEGE OF ENGINEERING

Note:

After constructing StringBuffer object

for each append() method call ,it increases capacity only one time .

capacity=2*(capacity+1)

if it exceeds that range, It takes capacity=2*(capacity+1)+characters length

example1:

StringBuffer s4= new StringBuffer();

s4.append("1234567890123456 7");

capacity=2*(16+1)=34

example2:

StringBuffer s5= new StringBuffer();

s5.append("12345678901234567890123456789012345678 90");

capacity=2*(16+1)=34(one increment) + length of remaining characters.

=34+ 6(remaining 6 characters)=40

```
class StringBufferCapLen {
    public static void main( String args[] )
    {
    StringBuffer b = new StringBuffer();
    b.append("hello");
    System.out.println(b.capacity());
    }
    }
```

```
class StringBufferCapLen {
    public static void main( String args[] )
    {
    StringBuffer b = new StringBuffer();
    b.append("1234567890123456");
    System.out.println(b.capacity());

    }
    }
```

```
class StringBufferCapLen {
    public static void main( String args[] )
    {
    StringBuffer b = new StringBuffer();
    b.append("12345678901234567");
    System.out.println(b.capacity());

    }
    }
```

```
class StringBufferCapLen {
    public static void main( String args[] )
    {
    StringBuffer b = new StringBuffer();
    b.append("12345678901234567123456789012345 67");
    System.out.println(b.capacity());

    }
    }
```

```
class StringBufferCapLen {
    public static void main( String args[] )
    {
    StringBuffer b = new StringBuffer();
    b.append("12345678901234567123456789012345671234567890");
    System.out.println(b.capacity());

    }
    }
```

```
class StringBufferCapLen {
    public static void main( String args[] )
    {
    StringBuffer b = new StringBuffer();
    b.append("12345678901234567123456789012345671234567890");
    System.out.println(b.capacity());
    b.append("12345678901234567123456789012345671234567890");
    System.out.println(b.capacity());
    b.append("12345678901234567123456789012345671234567890");
    System.out.println(b.capacity());
    }
    }
```

```
class StringBufferCapLen {
    public static void main( String args[] )
    {
    StringBuffer b = new StringBuffer();
    b.append("12345678901234567123456789012345671234567890");
    System.out.println(b.capacity());
    b.append("123456789012345671234567890123456712345678901234");
    System.out.println(b.capacity());
    b.append("123456789012345671234567890123456712345678901234");
    System.out.println(b.capacity());
    }
    }
```

JDK 1.7 Features

Mahindra École Centrale
COLLEGE OF ENGINEERING

1.Switch argument and label can be strings.

2. Binary representation with 0b.

3.Underscore between integer number.

1.Switch argument and label can be String type.

Mahindra École Centrale
COLLEGE OF ENGINEERING

```
class Sample{
public static void main(String... a){
switch("equal"){
case "not equal":
    System.out.println("!="); break;
case "equal":
    System.out.println("=="); break;
default:
 System.out.println("no string");}
}}
```

2. Binary representation with 0b.

Mahindra École Centrale
COLLEGE OF ENGINEERING

```
class Sample{
public static void main(String... a){
int b=0b111;  //zerob
 System.out.println(b);
}
}
```

3.Underscore between integer number

Mahindra
École Centrale
COLLEGE OF ENGINEERING

```
class Sample{
public static void main(String... a){
int b=98_48_123;
System.out.println(b);
}
}
```

Abstract Classes

abstract Classes

Mahindra École Centrale
COLLEGE OF ENGINEERING

- What is an abstract class

  A class which is not complete is an abstract class.

  Not complete means any method which doesn't have body.

- Limitation of abstract class

  We cant create object(s) to the abstract class. (instantiation not possible)

- How can we specify class is abstract

  Using abstract keyword

- Need of abstract class

  To standardized super class methods/behaviors to the sub class.

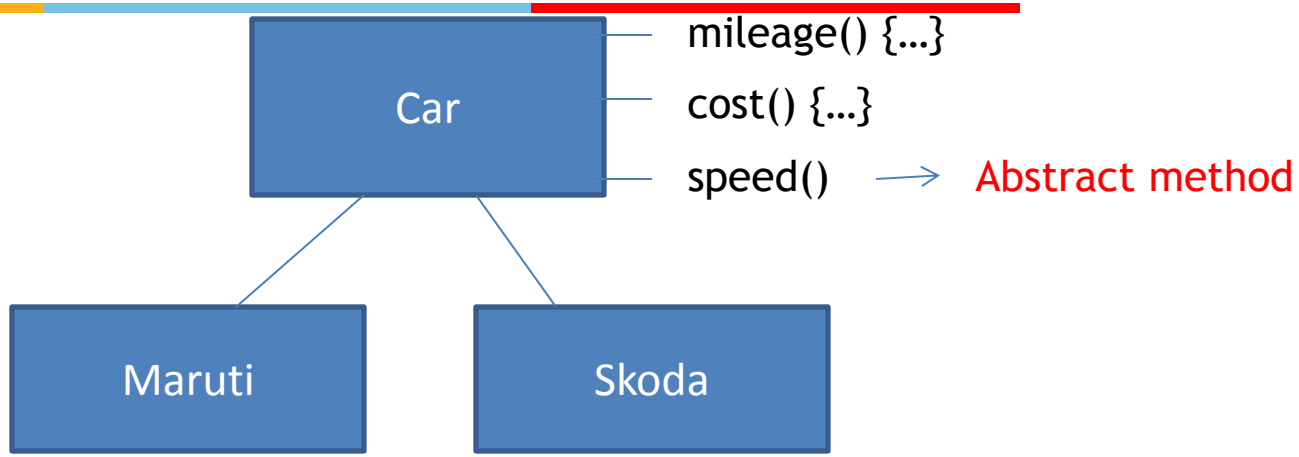
  Standardized means given specification has to follow / implement in sub classes based on their requirements.

- How can we restrict object creation to any class.
  Using abstract class.

- Even we don’t have abstract methods in the class, we can declare class as abstract.

- What is the purpose of concrete class when it is declared as abstract
  To promote “is-a” relationship.

Program for abstract class



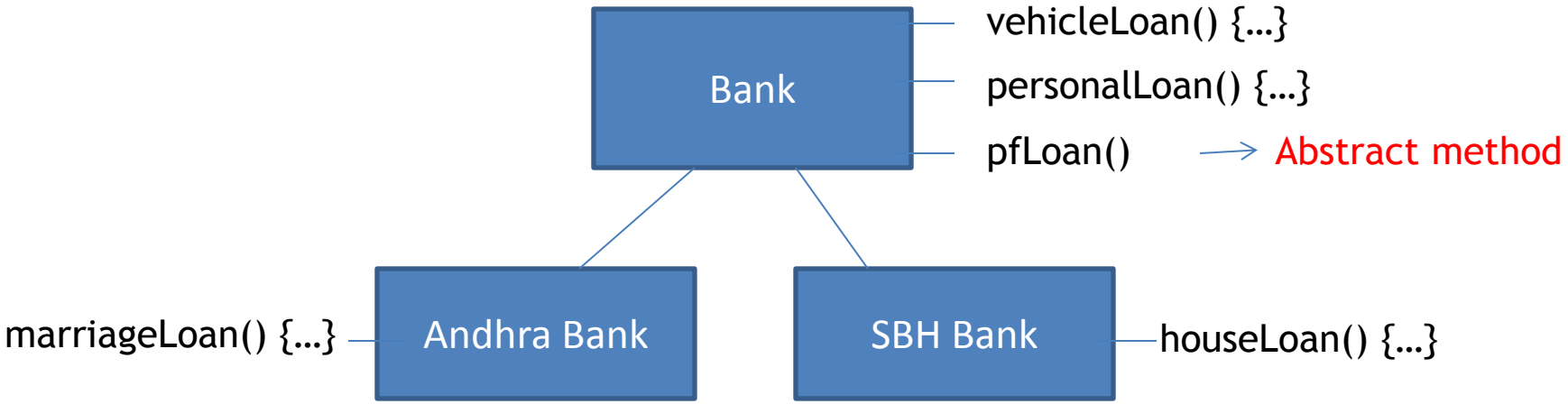

```
abstract class Car {
void mileage() { … }
void cost() { … }
abstract void speed () ;
}

class Skoda extends Car {
void speed() { … }
}
```

```
class Maruti extends Car
{

 void speed() { … }
 }
```

Examples


Bank
vehicleLoan() {...}
personalLoan() {...}
pfLoan()
Abstract method
marriageLoan() {...}
Andhra Bank
SBH Bank
houseLoan() {...}

Abstract class Vs Concrete class

Mahindra École Centrale
COLLEGE OF ENGINEERING

Abstract class	Concrete class
We cant create objects	We can create objects
It can have combination of concrete methods and abstract methods.	It can have only concrete methods
It can have constructors	It can have constructors
Inheritance is needed	Inheritance is not compulsory
Should declare with a keyword “abstract”	Need not declare with any keyword