Fundamentals of Object Oriented Programming

CSN- 103

Dr. R. Balasubramanian

Associate Professor

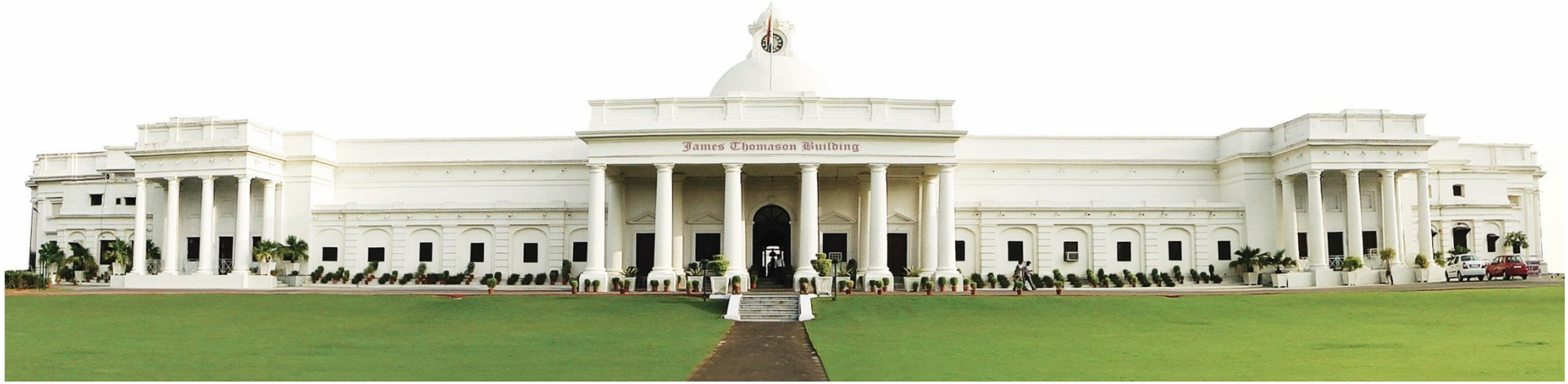
Department of Computer Science and Engineering

Indian Institute of Technology Roorkee

Roorkee 247 667

balarfcs@iitr.ac.in

https://sites.google.com/site/balaiitr/



Ways to achieve Abstraction

- There are two ways to achieve abstraction in java
  - Abstract class (0 to 100%)
  - Interface (100%)

Abstract class in Java

- A class that is declared as abstract is known as **abstract class**. It needs to be extended and its method should be implemented. It cannot be instantiated.

- Abstract class Syntax:

```
abstract class A{ }
```

Abstract method

- A method that is declared as abstract and does not have implementation is known as abstract method.

- **Example abstract method**

```
abstract void printStatus();//no body and abstract
```

Example of abstract class that has abstract method

```java
abstract class Bike{
  abstract void run();
}

class Honda4 extends Bike{
void run(){System.out.println("running safely..");}

public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

stdout

```
running safely..
```

- https://ideone.com/0kII3U

```java
abstract class Bike{
  abstract void run();
}

class Honda4 extends Bike{
void run(){System.out.println("running safely..");}

public static void main(String args[]){
 Bike obj = new Bike();
 obj.run();
}
}
```

compilation info

```
Main.java:9: error: Bike is abstract; cannot be instantiated
 Bike obj = new Bike();
            ^
1 error
```

stdout

```
Standard output is empty
```

https://ideone.com/vhlZOm

Understanding the real scenario of abstract class

```java
abstract class Shape{
abstract void draw();
}
//In real scenario, implementation is provided by others i.e. unknown by end user
class Rectangle extends Shape{
void draw(){System.out.println("drawing rectangle");}
}

class Circle1 extends Shape{
void draw(){System.out.println("drawing circle");}
}

//In real scenario, method is called by programmer or user
class TestAbstraction1{
public static void main(String args[]){
Shape s=new Circle1();
s.draw();
}
}
```

stdout

```
drawing circle
```

- https://ideone.com/2qjQAt

Abstract class having constructor, data member, methods etc.

```java
//example of abstract class that have method body
 abstract class Bike{
   Bike(){System.out.println("bike is created");}
   abstract void run();
   void changeGear(){System.out.println("gear changed");}
 }

 class Honda extends Bike{
 void run(){System.out.println("running safely..");}
 }
 class TestAbstraction2{
 public static void main(String args[]){
  Bike obj = new Honda();
  obj.run();
  obj.changeGear();
 }
}
```

stdout

```
bike is created
running safely..
gear changed
```

- https://ideone.com/ypzPfe

Another example of abstract class in java

```java
abstract class Bank{
abstract int getRateOfInterest();
}

class SBI extends Bank{
int getRateOfInterest(){return 7;}
}
class PNB extends Bank{
int getRateOfInterest(){return 8;}
}

class TestBank{
public static void main(String args[]){
Bank b=new SBI();//if object is PNB, method of PNB will be invoked
int interest=b.getRateOfInterest();
System.out.println("Rate of Interest is: "+interest+" %");
}}
```

stdout

```
Rate of Interest is: 7 %
```

- https://ideone.com/MklYyq

Abstract Class

- ***If there is any abstract method in a class, that class must be abstract.***

```
class Bike12{
abstract void run();
}
```

- compile time error

Final in inheritance

Java Final Keyword

- ⇨ Stop Value Change
- ⇨ Stop Method Overridding
- ⇨ Stop Inheritance

final float pi=3.14;

pi = pi+2;

Java final method

```java
class Bike{
  final void run(){System.out.println("running");}
}

class Honda extends Bike{
   void run(){System.out.println("running safely with 100kmph");}

   public static void main(String args[]){
   Honda honda= new Honda();
   honda.run();
   }
}
```

Terminal

```
sh-4.3$ javac Honda.java
Honda.java:6: error: run() in Honda cannot override run() in Bike
   void run(){System.out.println("running safely with 100kmph");}
        ^
  overridden method is final
1 error
sh-4.3$
```

- https://ideone.com/24Ghre

Java final class

If you make any class as final, you cannot extend it.

```java
final class Bike{}

class Honda1 extends Bike{
  void run(){System.out.println("running safely with 100kmph");}

  public static void main(String args[]){
  Honda1 honda= new Honda1();
  honda.run();
  }
}
```

Terminal

```
sh-4.3$ javac Honda1.java
Honda1.java:3: error: cannot inherit from final Bike
class Honda1 extends Bike{
                     ^
1 error
sh-4.3$
```

https://ideone.com/krEcRi

Is final method inherited?

- Yes, final method is inherited but you cannot override it.

```java
class Bike{
  final void run(){System.out.println("running...");}
}
class Honda2 extends Bike{
   public static void main(String args[]){
       new Honda2().run();
       //Bike b1;
       //b1=new Honda2();
       //b1.run();
   }
}
```

Terminal

```
sh-4.3$ javac Honda2.java
sh-4.3$ java Honda2
running...
sh-4.3$
```

- https://ideone.com/qulQhx

Example of blank final variable

```
class Student{
int id;
String name;
final String PAN_CARD;
...
}
```

Can we initialize blank final variable?

- Yes. but only in constructor

```java
class Bike10{
  final int speedlimit;//blank final variable

  Bike10(){
  speedlimit=70;
  System.out.println(speedlimit);
  }

  public static void main(String args[]){
    new Bike10();
 }
}
```

Terminal

```
sh-4.3$ javac Bike10.java
sh-4.3$ java Bike10
70
sh-4.3$
```

- https://ideone.com/HP3VCJ

static blank final variable

```java
class A{
  static final int data;//static blank final variable
  static{data=500;}
  public static void main(String args[]){
    System.out.println(A.data);
 }
}
```

Terminal

```
sh-4.3$ javac A.java
sh-4.3$ java A
500
sh-4.3$
```

https://ideone.com/2s64Ew

http://ideone.com/gtBUcc

(if you don’t put static)

final parameter

</> source code

```
class Bike11{
   int cube(final int n){
     n=n*n*n;  //can't be changed as n is final
     return n;
   }
   public static void main(String args[]){
     Bike11 b=new Bike11();
     b.cube(5);
  }
}
```

https://ideone.com/DBopyH

input Output

Compilation error #stdin compilation error #stdout 0s 0KB

```
Main.java:3: error: final parameter n may not be assigned
    n=n*n*n;  //can't be changed as n is final
    ^
1 error
```

- Packages and Interfaces

Java Package

- A **java package** is a group of similar types of classes, interfaces and sub-packages.

- Package in java can be categorized in two form, built-in package and user-defined package.

- There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

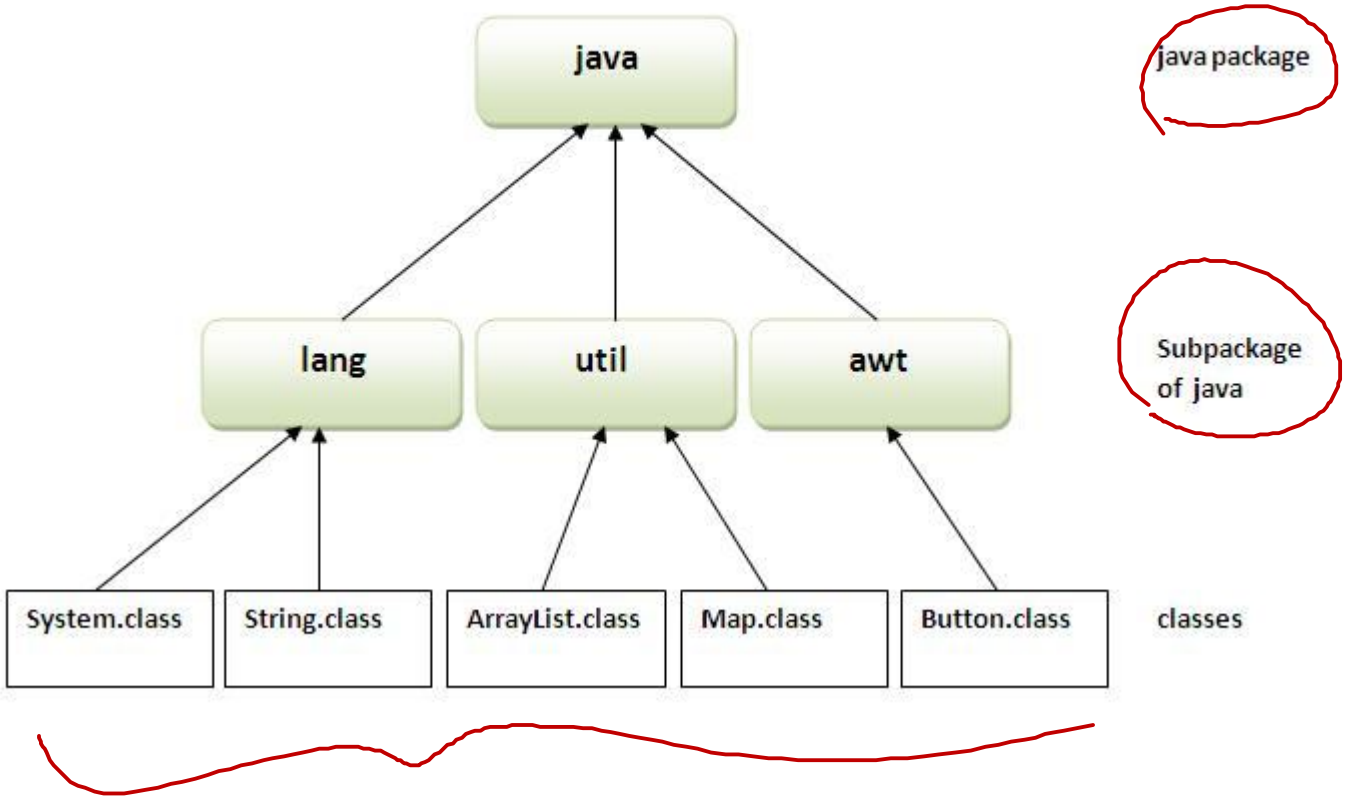
- Here, we will focus on user-defined packages.

Advantages of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.


java
lang
util
awt
System.class
String.class
ArrayList.class
Map.class
Button.class
java package
Subpackage of java
classes

Simple Example of Java Package

```java
package mypack;
public class Simple{
 public static void main(String args[]){
    System.out.println("Welcome to CSN-103, IIT Roorkee to learn the concept of package");
   }
}
```

Terminal

```
sh-4.3$ javac -d . Simple.java
sh-4.3$ java mypack.Simple
Welcome to CSN-103, IIT Roorkee to learn the concept of package
sh-4.3$
```

- http://goo.gl/jBRYHT

How to compile java package

- In Terminal
- javac -d directory javafilename (in general)
- javac -d . Simple.java (particular example)
- The -d switch specifies the destination where to put the generated class file. You can use any directory name like /home (in case of Linux), d:/abc (in case of windows) etc. If you want to keep the package within the same directory, you can use . (dot).