Programming with Java for BeginnersBineet Sharma

Class, Exceptions, Scope

Class and Object Series

Assumptions & ExpectationsClass and Object Series

- Assumptions
 - Method and Class series
- Expectations
 •Understand classes

Objectives Class and Object Series

- Class
- Error handling
- · Scope and life time of variables

Class and Objects:

Class and Object Series I

Java is an object oriented programming language Essential to learn about objects

Real-world objects: computer, desk, dog, car

All real-world objects have state and behavior

A dog's states are his/her color, breed, name A dog's behaviors are barking, fetching, wagging

Identifying state and behavior of real-world objects leads to thinking in OO programming

Class and Objects:

Class and Object Series I

Software objects consist of state and behavior

int myAge = 12; Literally myAge is an object whose state is value 12. The behavior of myAge is: it can be added, subtracted, compared etc.

Concept of objects are applied to new type of objects created by programmers where built in objects (data types) are not sufficient to solve the need.

For example: String object (character arrays are not sufficient to provide the functionality)

Defining Classes:

Class and Object Series I

- User stories (requirement gathering)
- · Identify nouns to make up your class
- Organize classes to solve user stories
- · Restrict access to data within class
- Use visibility modifiers in methods for accessibility to clients (of class)
- Pass data to/from methods as parameters
- Create instance variables for class, built in types and use locally or pass as parameters
- Create helper methods as utility

Defining Classes:

Class and Object Series I

- User stories (requirement gathering)
- Identify **nouns** to make up your class
- Verb/Adj make up behavior and \underline{state}
- Organize classes to solve user stories

John loves drawing boxes. He wants to draw boxes of different <u>shape</u>, <u>size</u> and <u>colors</u>. He does not know what <u>size</u>, <u>shape</u> or <u>color</u> of box he wants to draw beforehand but he would know once he starts drawing.

Defining Classes:

Class and Object Series I

• Identify nouns, verbs and adjectives

Nouns: John (<u>client</u>), He, <u>Box</u>

Verb: <u>Draw</u>, drawing

Adjective: Shape, size, color

Defining Class: Class and Object Series I

Class: A blue print to create an individual object. It describes the characteristics of similar objects. Small box or big box: both of them are a Box

```
public class Box {
  int width = 20;
  int height = 10;

  void changeHeight() {40};
  void changeWidth() {20};

  void Draw();
```

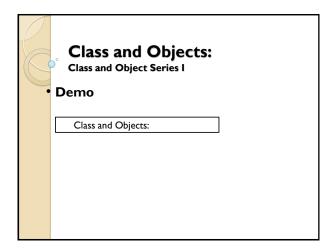
Class and Objects: Class and Object Series I Object's state is stored in fields (variables) it could be primitive data type or another object public class Box { int width = 20; int height = 10; String hLineSymbol = "-"; String vLineSymbol = "|"; ... }

Class and Objects:

Class and Object Series I

Object's behavior is defined by **methods** (functions) it operates on internal state. Hides details. Providing OOP Encapsulation.

```
void changeHLSymbol(String hLine)
{
    hLineSymbol = hLine;
}
void changeVLSymbol(String vLine)
{
    vLineSymbol = vLine;
}
```



Defining Class:

Class and Object Series II

public void Draw();

- Restrict access to data within class
 Use visibility modifiers in methods for accessibility to clients (of class)
 Create helper methods as utility
- public class Box {
 private int width = 20;
 private int height = 10;

 private void trimHeight() {};

 public void changeHeight() {40};

Using Class:

Class and Object Series II

Object: A BIG BOX or a small box is an instance (object) of a Box blue print (class)
Object is run time entity consisting of data (state) and responds to messages (behavior)

```
import MyUtility.*;
public class FirstJavaHello {
    public static void main(..){
        Box bigBox = new Box();
        bigBox.drawBox();
    }
}
```

Using Class:

Class and Object Series II

An object is an instance of its class (blue print), and the process of creating a new object is called instantiation.

bigBox is **instantiated** from Box class **new** operator is used to instantiate a new object

```
import MyUtility.*;
public class FirstJavaHello {
   public static void main(..){
      Box bBox, sBox; //declr varble
      bBox = new Box(); //instantiate
      bBox.drawBox(); //use
   }
```

Using Class:

Class and Object Series II

bBox is a reference to an box object (bBox is an interface not real object --which is in memory).

A box object keeps track of height and width
What is the value of height and width of new object?
Use accessor methods

```
Box bBox, sBox; //declr varble bBox = new Box(); //instantiate

System.out.println(bBox.findHeight()); System.out.println(bBox.findWidth()); System.out.println(bBox.findVLSymbol()); System.out.println(bBox.findHLSymbol());
```

Using Class: Class and Object Series II

bBox is a reference to an box object A box object keeps track of height and width How do you change these values? Use mutator methods:

```
Box bBox, sBox;
                  //declr varble
bBox = new Box(); //instantiate
bBox.changeHeight(8);
bBox.changeWidth(8);
bBox.changeHLineSymbol("*");
bBox.changeVLineSymbol("*");
```

Using Class:

Class and Object Series II

Always write a special method called: toString as well

```
public String toString()
return "\nName: " + boxName +
      "\nHeight: " + height +
      "\nWeidth: " + width +
      "\nH Line Sysmbol: " +hLineSymbol +
      "\nV Line Symbol: " + vLineSymbol +
      "\n\n";
```

Using Class:

Class and Object Series II

You can directly call the **toString** method for that

However, there are other situations where the method toString is implicitly called.

For instance:

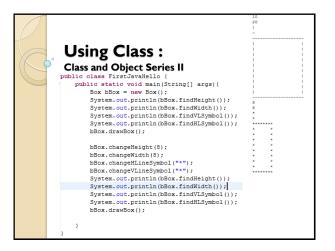
- toString is called implicitly when a Box object is concatenated with a string.
- toString is called when Box object is in an argument to the method **println**

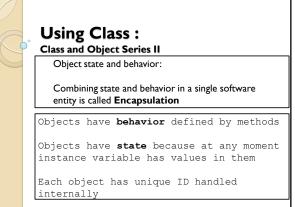
Using Class:

Class and Object Series II

Example of implicit use of toString method

```
String str;
str = "This is Johns Big Box: \n" +
                     bigBox.toString();
//this is equivalent
str = "This is Johns Big Box: \n"+bigBox;
System.out.println(smallBox.toString());
//this is equivalent
System.out.println(bigBox);
```





Using Class:

Class and Object Series II

Object state and behavior:

Combining state and behavior in a single software entity is called **Encapsulation**

```
public class Box {
   private int width = 20;
   ...
   private String vLineSymbol = "|";
   ...
   void changeHLSymbol(String hLine) {};
   ...
}
```

Using Class:

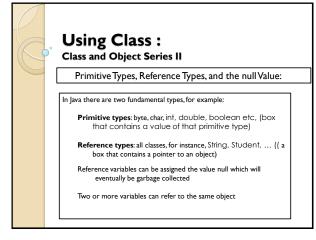
Class and Object Series II

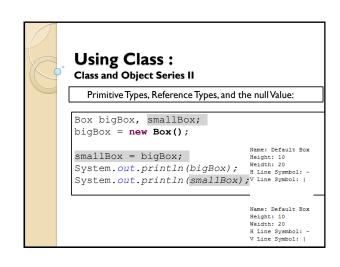
JVM, Computer Memory and Your Program:

Memory holds:
 compiled class templates(always)
 variables that refer objects and
 object themselves (come and go)
 - in first instantiation,
 disappears when not needed

JVM keep track of object references

Unreferenced objects are deleted via
 garbage collection





Using Class:
Class and Object Series II

Primitive Types, Reference Types, and the null Value:

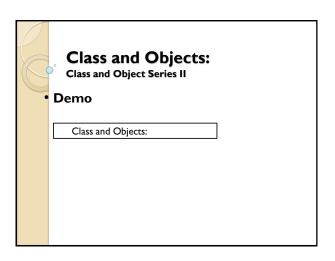
smallBox = bigBox;

//you can break the connection by assigning null to it smallBox = null;

//you can only assign null to objects not //primitive variables

//int i;

//i = null; //cant be done



Errors in Java Program:

Class and Object Series III

- Syntax errors: compiler will report this error
- Semantic error: logical error compiler has no clue
- · Runtime errors: logical error which crashes your prog
 - I. Divide by zero
 - 2. Subscript out of range for arrays
 - 3. Stuffing wrong data type
 - 4. Calling a function on null object
- How to tackle them?

Exception in thread "main" <u>java.lang.NullPointerException</u> at FirstJavaHello.main(<u>FirstJavaHello.java:508</u>)

Errors in Java Program:

Class and Object Series III

- •How to tackle them?
 - 1. Syntax errors: With help of compiler and document
 - 2. Logical errors: by debugging and not doing sloppy programming
 - 3. Runtime errors by using exception handling

Exception in thread "main" java.lang.NullPointerException at FirstJavaHello.main(FirstJavaHello.java:508)

Errors in Java Program:

Class and Object Series III

Errors in Java Program:

Class and Object Series III

Using the function which checks error for us:

How do you improve this to use for all numbers?

```
Box bigBox, smallBox;
bigBox = new Box();

int boxSize = getValidInteger();
bigBox.changeHeight(boxSize);

bigBox.changeWidth(getValidInteger());
```



Exceptions in Java Program:

Class and Object Series III

When your program crashes for some situation which JVM does not know how to handle (divide by zero or subscript out of range or null pointer exception), JVM will create a special object called **exception** and throw.

As a programmer it is up to you to catch that exception object and handle the situation relatively gracefully

Exception in thread "main" <u>java.lang.NullPointerException</u> at FirstJavaHello.main(<u>FirstJavaHello.java:508</u>) Something horrible happen while drwing smallbox: java.lang.NullPointerException

Using Class:

Class and Object Series III

Introducing Exception:

Class and Objects:

Demo

Exception Handling:

Class Construct: Class and Object Series IV

Most classes have similar constructs

- A name with some modifiers
- Some instance variables
- One or more method describing how to initialize the object (**Constructors** method)
- A constructor is a special method with the same name as the class
- More methods which signifies how your object responds to messages

Class Construct:

Class and Object Series IV

Most classes have similar constructs

Class Construct:

Class and Object Series IV

Most classes have similar constructs
Constructor is called when you use **new** operator
Used for initializing the state of newly
instantiated object

Can have multiple constructors **Default Constructor** has empt

Default Constructor has empty body If you don't provide JVM provides one DC If you have multiple constructor can chain them

public Box() {}
public Box(int w, int h, String hLine,
String vLine, String boxName) {}

Life Time and Scope of Variables:

Class and Object Series IV

Class has two main parts: Instance variables and Methods

When a new object is created it will have its own new set of instance variables (Global Variables) (common pool of data for all methods)

Object: contains data and responds to messages

Class: has template to create object and has code

When a method executes – it does for an object it has complete access of instance variable and local variable

Scope of Variables:

Class and Object Series IV

Scope is region of program where the variable is visible

Scope of parameter and local variable is restricted to the methods

Scope of instance variable (global variable) is all methods of the class

Compiler will flag it as error if you go out of scope

Scope of Variables:

Class and Object Series IV

Scope of Variables:

Class and Object Series IV

```
Block scope {}

{
    int myInt = 50;
}

{
    int myInt = 70;
}
```

Life Time of Variables:

Class and Object Series IV

The time period during which the variable can be used

Local variables: Single execution of method **Formal parameter:** Single execution of method (new value for next call)

Instance (global) Variable: Available during life time of object

Recycling (**duplicating**) the variable names: Is allowed because of scope resolution will take care of the issue

Life Time of Variables:

Class and Object Series IV

Good usage of:

- Instance variable
 - Be careful not to use it for other purposes
- Parameter
- · Local variable
 - Do not use to remember sate of object

Class Hierarchy in Java:

Class and Object Series IV

Java has hierarchy of classes

Object is the base of this hierarchy

A class immediately after another class is called subclass (child) of its superclass (parent)

All classes in Java has one parent and many children (except Object class)

Class Hierarchy in Java: Class and Object Series IV

Java has hierarchy of classes

A new class in the hierarchy will inherit commonality of its parent class (superclass)

You use extend keyword to extend the base class

If you omit extend, it automatically inherits from Objects

Class Hierarchy in Java:

Class and Object Series IV

Inheritance:

Different kinds of objects have commonality and differences: Some additional features

Mountain bikes have gears, while mountain bikes also have two wheels like road bike.

Specialized objects inherit the commonality from other

subclass inherits commonality from superclasses class SolidBox extends Box{ ...}

Class and Objects:

Class and Object Series IV

Demo

Write your class:



- Class
- Error handling
- Scope and life time of variables