STRINGS

Implementing strings as built-in objects allows JAVA to provide full compliment of features that makes string handing convenient. for ex:- JAVA has method to compare two strings, search for sub-string, concatenate 2 strings & change the case of letters within a string. Also string objects can be constructed a no of ways making it easy a string when needed.

The String, Stringbuffer & Stringbuilder classes are defined in JAVA. lang package. Thus they are available to all programs automatically.

STRING CONSTRUCTORS

The string class supports several constructors to create an empty string. You call the default constructor.

```
String S = new String();
```

This will create an instance of string with no characters in it. A string class provides a variety of constructors to handle. We want to create strings that have initial values.

To create a string initialised by array of characters

```
string ( char, char[])

Char Char[] = {'a', 'b', 'c'};
String S = new string (char);
```

```
Class Makestring
{   p. s. v. main( String args[])
    {   char C[]= { 'J', 'A', 'V', 'A'};
        String S1= new String(c);
```

String S2= new string (S1); S.O.pln (S1); S.O.ph(S2);
}}

Even though JAVA's char type uses 16 bits to represent the basic unicode set. The typical format of string on the internet uses arrays of 8 bit bytes constructed from ASCII character set because 8 bit ASCII strings are common, the String class provides constructors that initialises the string when given a byte array

String (byte asciichars [])
String (byte asciichars [], int startindex, int numchars)

here asciichars [] specify the array of bytes. The second form allows you to specify a sub-range. In each of these constructors byte to character conversion is done by using the default character & coding of the platform.

13th March, 19

Class substring
{ p.s.v.main
{ byte ascii[] = {65,66,67,68,69,70};
String S1 = new String (ascii);
S.O.pln(S1); (a,b,c,d,e,f)
String S2 = new String (ascii(2,3));
S.O.ph (S2); (c,d,e)
}
}

Construct a string from a subset of character array.

STRING LENGTH

The length of a string is the no. of characters that it contains to obtain this value we call the length() method.

```
int length()
Char chars[] = { 'a', 'b', 'c' };
String s = new String (chars);
S.O.pln ( s. length());
```

```
String age = "9";
String s = "He" + age + "years old";
s.o.pln (s)*;
```

→ Concatenation

STRING CONVERSION AND THIS METHOD

tostring() valueof()

When JAVA converts data into it's string representation during concatenation it does so by calling one of the overloaded versions of string conversion method value of() defined by String. valueof() is overloaded for all the simple types & for type object. for the simple types value of() returns a string that contains the human readable equivalent of value with which it is called for objects value of calls the tostring() method on the object. Every class implement tostring because it is defined by object however the default implementation of tostring is seldom sufficient for most imp. classes that you create, you will want to override tostring & provide your string representation.

8th March 19

To implement tostring simply return a string object that contains the human readable string

that appropriately describes the object of the class by overriding toString for classes that you create. You allow them to fully integrate into JAVA's programming environment for ex:- They can be used in print() & println() statements & in concatenation expressions

```
Class Box
{   double width;
    double height;
    double depth;
Box (double w, double h, double d)
{   width = w;
    height = h;
    depth = d;
}
public String toString()
{   return "Dimensions are" + width + "by" + depth
        + "by" + height + ".";
    }
Class TestingDemo
{   p.s.v.main  Box
    {   Box b = new( 10,12,14);
        String S = "Box b:" + b;
        SOP(b);
        SOP(S);
    }
}
```

width over
without

CHARACTER EXTRACTION METHODS

1. ~~A String class~~ CharAt()

To extract a single character from a string you can refer directly to an individual character via CharAt() method.

Char CharAt(int where);

here where is the index of the character that you want to obtain.

```
Char ch;
ch = "abc".charAt(1);   ⇒ b.
```

2. getChars()

If you need to extract more than one character at a time you can use the getchars() method.

```
void getChars(int sourcestart, int sourceend,
    Char target[], int targetStart);
```

→ sourcestart specifies the index of begining of substring.

→ sourceend specifies an index that is one past the end of desired substring. Thus the substring contains the characters from sourcestart to sourceend − 1. The array that will recieve the characters is specified by target[].

→ The index within target[] at which the substring will is copied is passed in targetstart

3. getBytes()

This method uses the character to byte conversion. getBytes() is useful when you are exporting a string value.

4. ToCharArray()

If you want to convert all the characters in a string object into a character array we can

use this method. It returns an array of characters for the entire string.

- STRING COMPARISON METHODS

1. Equals()

return type = boolean

equal(object str())

here str() is the string object being compared with the invoking string object

2. equalsIgnoreCase()

19th March, 19

- DATA CONVERSION USING ValueOf()

i) The valueOf() method converts data from it's internal format into human readable format. It is a static method that is overloaded within string for all JAVA's inbuit types so that each type can be ~~overloaded~~ converted properly into string.

ii) ValueOf() is also overloaded with type object so when object of any class type you create can also be used as an argument valueof() is called when a string representation of another type of data is needed

- CHANGING CASE OF CHARACTERS / CHARACTER STRING

String toLowerCase();

String toUpperCase();

- STREAM

Java programs perform I/O through Streams: A Stream is an exception that either produces or consumes information. Stream is linked to physical

device by
Java defines 2 types of Streams:-

1 Bytestream
2 Character Stream

1. Bytestreams provide a convienient way for handling I/O of bytes. These are used when reading or writing binary data.

2. Characterstreams provide a convienent means for handling I/O characters. ~~For~~ Bytestreams are defined by using 2 class hierarchies at the top are 2 abtract classes:- Input Stream & Output Stream

2. Characterstreams are defined by using two class hierarchies. At the top are two abstract classes:- Reader & Writer.

1. The abstract classes Input & Output Stream define several key methods that the other Stream classes implement. Two of the most important are:- Read() & Write().

2. The abstract classes reader and writer define several key methods that other stream classes implement :- read() & write().

System.out refers to the standard output stream and System.in refers to standard input stream. System.err refers to standard error stream. System.in is an object of type input stream. System.out & System.err are the objects of type (output stream) print screen

READING CONSOLE INPUTS

In Java, console input is read accomplised by reading from System.in to obtain character based screen that is attached to the console wrap 'System.in' in a BufferedReader object.

BufferedReader (Reader InputReader)

Here inputreader is the stream that is linked to the instance of bufferedreader that is being created

Reader is an abstract class. One of it's concreate sub-classes is InputStreamReader which converts bytes to characters.

To read & Input from Console

```
Class BRead
{ p.s.v. main (String args[]) throws IOException
  { Char c;
BufferedReader br = new BufferedReader(new InputStream
                                    Reader(System.
S.O.P("Enter characters, 'q' to quit");
do{ c = (Char) br.read();
    SOP(c);
  } while (c != 'q');
 }
}
```

To obtain an input stream reader object i.e. linked to System.in

InputStreamReader (InputStream InputStream)

because System.in refers to the object of type InputStream, it can be used for InputStream

```
BufferedReader br = new BufferedReader(new InputStreamReader
                                    (System.in));
```

No 11th March, 14.

READING STRING

To read a string from the keyboard use the version of readline() that is a member of BufferedReader class.

String readline() throws IOException Package.

import java.i

```
Class BRReadLines
{   p.s.v.m throws IOException
{ BufferedReader br= new BufferedReader (new InputStream
                        Reader (System.in));

  String str;
  S.O.pln ("Enter lines of test");
  S.O.pln ("Enter 'STOP' to quit");
  do{
      Str= br.readline();
     S.O.pln (str);
  } while(!str.equals ("stop"));
}}
```

- APPLETS

It is imp. to state at the outset that there are 2 varieties of applets based on Applet (Class) The first are those that extend the applet class directly. These applets use abstract Window Toolkit (AWT) to provide graphical User Interface (GUI). The second type of applets are those that extend Swing class JApplet which extends Applet

Swing applet use the swing classes to provide the GUI. Swing offers a richer & often easier to use User Interface that does the AWT.

AWT based applets are direct sub-class of applet.
Applets are not stand alone programs instead they run within either a web-browser or an applet viewer. Unlike JAVA console application, execution of an applet does not begins at main().

- APPLET CLASS

Applet class encapsulates an applet. It defines a no. of methods. The most fundamental are those methods that control an applet's execution such as starting & stopping. These are the lifecycle methods

i) init()
ii) start()
iii) stop()
iv) destroy()

- APPLET INITIALISATION & TERMINATION

When an applet begins the following methods are called in sequence:-

i) init():- The init() method is the first one to be called. This is where applet initialisation is performed. This method is called only once during the execution of the applet.

ii) start():- The start() method is called after init(). It is also called to restart an applet after it has been stopped. Whereas init() is called once, the first time an applet is loaded, start() is called each time an applet document is displayed. So if a user leaves a web page & comes back the applet resumes execution at start().

iii) paint():- The paint() method is called each time an AWT based applets output must be re-drawn. Paint() is not specified by applet but it is inherited from Component (class).

iv) stop():- This method is called when web-browser leaves HTML document containing applet; when it goes to another page. Eg:- when stop() is called applet is probably running an important use of stop() is to suspend thread that doesn't need to run when applet is not visible.

v) destroy():- This method is called when environment determines that applet needs to be removed completely from memory at this time any resources used by applet should be released.

```
import Java.awt.*;
import.Java.applet.*;
/* <Applet code = "AppletSkel" width = 300, height=100;
   </Applet> */

public class AppletSkel extends Applet
{   public void init()
    {   // Initialisation
    }
    public void start()
    {   // start or resume execution
    }
    public void stop()
    {   // Suspends execution
    }
```

26th March, 19

```
public void destroy ()
{ // perform shutdown activities }
public void paint ( Graphics g)
{ g.drawString ("AWT based Appletskeleton", 20, 20); }
}
```

When developing applets you can use either the Applet or other object tag to easily view or test Applet. To do so simply include a comment at head of JAVA source code file that contain applet or object tag. This way your code is doumented with necessary HTML statements needed by your applet & you can test compiled Applet by starting applet viewer with your JAVA source code file specified as the target.

27th March, 19

AWT

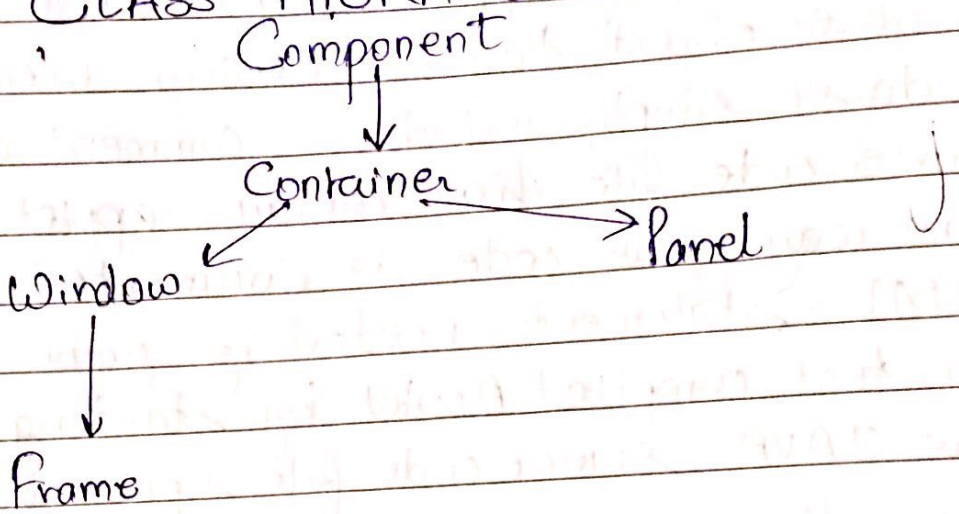
The abstract window toolkit was JAVA's first GUI framework. It contains numerous classes & methods that allow you to create windows & simple controls. AWT supports following type of controls -

i) Label
ii) Pushbuttons
iii) Checkboxes
iv) List
v) Choicelist
vi) Scrollbars
vii) Text Editing

The AWT defines windows acc. to a class hierarchy that adds functionality & specificity with each level. The two most imp. window related classes are Frame & Panel. Frame encapsulates a top level window and is typically used to create what would be though of as standard app window.

ii) Panel provides a container to which basic componen are added. Panel is also super-class of Applet much of functionality of frame. Frame & Panel if derived from their parent classes:-

- AWT CLASS HIERARCHY



★ Component is an abstract class that encapsulates all of the attributes of a visual component except for menus. All user interface elements that are displayed on the screen that interact with user are sub-classes of component.

★ Container class is sub-class of component. It has additional methods that allow other component objects to be nested within. Other container objects can be stored inside of a container. A container is responsible for laying out any componer that it contains.

★ Panel class is concrete sub class of container. Panel may be though of as recursively nestable concrete sub-component. Other Components can be added screen to panel object by it's add() method.

Use of frames

Page No.:
Date:
YOUVA

* The window class creates a top level window. A top level window is not contained with any other object.

* Frame encapsulates window. It is a sub-class of window & has a title bar, menu bar, borders & resizing corners.

- CANVAS COMPONENT

Derived from component, canvas encapsulates a blank window upon which you can draw.

The AWT includes several methods that support graphics. All graphics are drawn relative to a window. The origin of each window is at the top left corner and is at (0,0). Co-ordinates are specified in pixels. All output to a window takes place through a graphics context. A graphics context is encapsulated by graphics class. The two ways in which a graphics context can be obtained:

i) It is passed to a method such as paint() or update() as an argument ii) It is returned by getgraphics() method. Among other things the graphics class defines a no. of methods that draws various types of objects such as lines, rectangles & arcs.

1st April, 14

The paint() method is called each time. An AWT based applications output must be re-drawn. This situation can occur for several reasons. For example:- Programs window may be overriden by other window & then uncovered of the window may be minimised or restored

- Demonstrate the paint() method.

```
import Java.awt.event.*;
import Java.awt.*;

public class graphicsdemo extends Frame
{   public graphicsdemo()
    {   addwindowlistener(new windowadapter(){
        {   public void windowclosing(windowevent w
                System.exit(0);
            });
    }
```

Anonymous innerclass

It is an anonymous inner class to handle windowclose event.

```
public void paint(Graphics g)
{ // drawing lines
    g.drawline(20,40,100,90);
    g.drawline(20,90,100,40);
    g.drawline(40,45,250,80);
  // draw rectangles   (co-ordinates, height, width)
    g.drawRect(20,150,60,50);
    g.fillRect(110,150,60,50);
    g.drawroundRect
    g.drawroundRect fillRect
    g.drawarc
    g.fillarc
    g.drawpolygon
}
```

ASSIGNMENT :- Explanation in comments

1. Reading & Writing Programs
2. To implement checkboxes

```
public static void main
{   Graphicsdemo obj = new Graphicsdemo();
        obj.setsize (new dimension (370, 700));
    obj.setTitle (" Applet Program");
    obj.setVisible (true);
}
```

- AWT CONTROL FUNDAMENTALS

i) Label ii) Pushbuttons iii) Checkboxes iv) List
v) Choicelist vi) Scrollbars vii) Text Editing.

All AWT controls are sub-classes of components. Although set of control provided by AWT is not particularly rich. It is sufficient for applications such as short utility program intended for your own use.

To include the control in a window you must add into window. You must first create an instance of desired control then add it to window by calling add().

8th April, 19

- TO PUT LABELS IN APPLET

```
import java.awt.*;
import java.awt.event.*;
public class labeldemo extends frame
{   public labeldemo()
    {   setlayout (new Flowlayout());   → automatic dimensions
        label one = new label("one");
        label two = new label ("two");
        Label three = new Label ("three");
        add one); //objects
        add (two);
        add (three);
```

```
addwindowlistener (new WindowAdapter ()
{
  public void WindowClosing (windowevent we
    {  system.exit(0);
    }
});
}  p. s. v. main
    { Labeldemo obj = new Labeldemo ();
      obj. setsize (new dimensions (300,300));
      obj. settitle ("Labels Window");
      obj. setvisible (true);
    }
}
```

The labels are organised in window left to right. This is handled automatically by the flow layout layout manager.

- TO PUT BUTTONS IN APPLET

```
import java. awt. *;
import java. awt. event. *;
public class buttondemo extends frame implements
                                    ActionListener
{
  String msg = "  ";
   Button Yes ; Button No, Maybe;
  public buttondemo ()
   {    setlayout (new Flowlayout ());
      Button Yes = new Button ("Yes");
        No = new Button ("No");
      Maybe = new Button ("Undecided");
    add(Yes); add(No);
        add (Maybe);
```

```
Yes.addActionListener(this);
No.addActionListener(this);
Maybe.addActionListener(this);
addwindowlistener(new WindowAdapter()
{public void windowClosing(windowevent ae)
    {System.exit(0);
    }
});
}
public void actionPerformed(ActionEvent ae)
{   String str = ae.getActionCommand();
    if(str.equals("Yes")
    {msg="You pressed Yes";}
else if(str.equals("No"))
    {msg="You pressed No";
    }
else (str.equals("Maybe")
    {msg="You pressed undecided";
    }
repaint();
}
public void paint(Graphics g)
{   g.drawString(msg,20,100);
}
P.S. void main
{   Buttondemo obj = new Buttondemo();
    obj.setsize(new dimensions(300,300));
    obj.setTitle("Button window");
    obj.setvisible(True);
}
}
```

TEXTFIELD TRY YOURSELF = 9th April, 19

- TO PUT TEXTAREA IN APPLET

```
import. java. awt. *;
import. java. awt. event. *;
public class textareademo extends frame
{   public textareademo ()
    {  setlayout ( new Flaylayout ());
       String val;
       val = " we are using Textarea in Applet";
TextArea text = new TextArea (val, 10, 30);
    add (text);
```

↳ very like decknd

```
addwindowlistener (new WindowAdapter ()
{ public void WindowClosing ( windowevent we
    {  system. exit(0);
    }
});
}
public static void main (String args [])
{   textareademo obj = new textareademo ();
    obj. setsize ( new dimensions (300, 300));
    obj. settitle ("TextArea");
    obj. setvisible (True);
}
```

General Syntax →

```
TextArea (String str, int numlines, intnumchar);
```

numlines specifies the height in lines of textarea and numchar specifies it's width in characters.

15th April, 19

- LAYOUT MANAGER

Layout manager automatically arranges your control within a window by using some type of algorithm. It is possible to layout JAVA controls by hand. It is very tideous to manually layout large no. of components ii) Sometimes the width & height information is not yet available when you need to arrange some control because the native toolkit component haven't been realised.

Each container object has a layout manager associated with it. A layout manager is an instance of any class that implements the 'LayoutManager' interface. The layout manager is set by setLayout() method. If no call to setLayout() is made then the default layoutManager is used. Whenever a container is resized the LayoutManager is used to position each of components within it.

```
void setLayout (LayoutManager & Layout obj);
```

If you wish to disable the LayoutManager & position components manually, we pass Layout obj as null. If you do this you will need to determine the shape & pos. of each component manually using the setBounds() method defined by component.

1. FlowLayout()

FlowLayout() implements a simple layout style which is similar to how words flow in a text editor. The direction of layout is governed by the container's component orientation property which by default is left to right & top to bottom. ∴ By default, the components are laid out line by line

Date:

begining at upper left corner. In all cases when a line is filled, layout advances to next line.

i) FlowLayout ():- It creates a default layout which centres components and leaves 5 pixels of space b/w each component.

ii) FlowLayout (int how):- Let's you specify how each line is alligned.
how:- FlowLayout.LEFT
FlowLayout.CENTER
FlowLayout.RIGHT
FlowLayout.LEADING
.TRAILING

iii) FlowLayout (int how, int horz, int vert):- Let's you specify the horizontal & vertical spacing b/w the components in an applet window.

2. BorderLayout ():- BorderLayout class implements a layout style that has four narrow, fixed width components at the edges and one large area in the centre. The four sides are referred as :- North, South, East & West. The middle area is called the centre. BorderLayout is default LayoutManager for the frame.