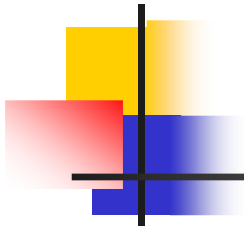
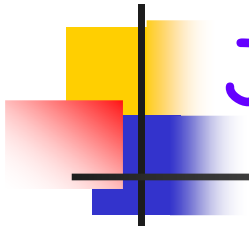


Lecture2: Programming basics in Java

- Data types
- Operators
- Input, output
- Control statements
  - if else
  - for
  - While
  - do
  - switch, case
- Functions
- Classes

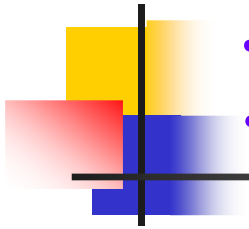


Java basic data types and operators

- Basic data types
  - byte, char, short, int, long, float, double, boolean
- Operators:
  - Arithmetic: +, -, *, /, %, ++, --
  - Logical: ==, !=, >, <, >=, <=, &&, ||, !, ?:
  - Bitwise: &, |, ^, <<, >>, ~

Java basic data types

Type	#bits	Default value	Min value	Max value
byte	8	0	-128	127
short	16	0	-32768	32767
int	32	0	-2147483648	2147483647
long	64	0L	-2^63	(2^63) - 1
float	32	0.0f		
double	64	\u0000		
Boolean	Not clearly defined	null	NA	NA
char	16	false	NA	NA



Input, Output

- Input
  - i = System.in.read( ) – read a character
    - i has the ASCII value of the character
  - BufferedReader br =
    new BufferedReader(new InputStreamReader(System.in));
  - String inValue = br.readLine( );
    - Reads a string from stdin
  - Others
    - We will (probably) see some more later.
- Output
  - System.out.println – print output to std. Output
    - E.g. System.out.println ("any string" + variable + …);
  - Others
    - There are others possible – we will see them later.

Control statements … if

```
if (<expr_1>)
{
    <body of if_expr_1>
}
else if(<expr_2>)
{
    < body of if_exp_2>
}
…
else /* default */
{
    …
}
```

Example-1

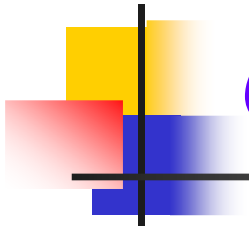
```
if (i > j)
    System.out.println ("i is larger\n");
```

Example-2

```
if (i > j)
    System.out.println ("i is larger\n");
else ( j > i)
    System.out.println ("j is larger\n");
```

Example-3

```
if (i > j)
    …
else if (i > k)
  …
else
 …
```



Control statements - for

- for (<start_expr>;
  <termination_cond>;
  <loop_increment>)

{

  <body_of_for>

}

Example-1 /* print 0 to 9 */

```
for (i = 0; i < 10; i++)
{
   System.out.println (i);
}
```

Example-2

```
for ( ;  ;)  /* infinite loop */
{
   /* do something */
}
```

Control statements - while

- Similar to for statement
- while (<while_cond>)

```
{
        <while_body>
}
```

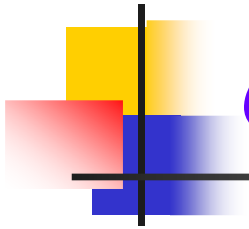
```
do
  {
        <body_of_do>
  } while (condition);
```

Example-1 /* print 0 to 9 */

```
i = 0;
while (i < 10)
{
        System.out.println (i);
        i++;
}
```

Example-2

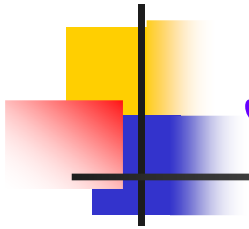
```
while (true) /* infinite loop */
{
        /* do something */
}
```



Control Statements - switch, case

```
switch (x)
{
    case val1:
        <val1_body>;
        break;
    case val2:
        <val2_body>;
        break;
    …
    default:
        <default_body>
}
```

```
int x = 2;
switch (x)
{
    case 1:
        function1( );
        break;
    case 2:
        function2( ); /* executed*/
        break;
    …
    default:
        default_function( );
}
```

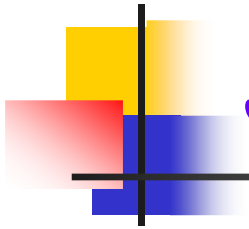


Java String

- Java String provides a rich collection of functions for
  - Comparison to other strings/sub-strings
  - Character insertion, retrieval
  - Replacing regular expressions
  - Conversion to upper/lower case

Java String … contd.

```
public final class String
{
    public int length( )
    public char charAt(int index)
    public void getChars(int srcBegin, int srcEnd, char dst[], int dstBegin)
    public boolean equals(Object anObject)
    public boolean equalsIgnoreCase(String anotherString)
    public int compareTo(String anotherString)
    public int compareToIgnoreCase(String str)
    public int hashCode( )
    public int indexOf(int ch)
    public String replace(char oldChar, char newChar)
    public boolean matches(String regex)
    public String replaceFirst(String regex, String replacement)
    public String replaceAll(String regex, String replacement)
    public String toLowerCase(Locale locale)
    public String toLowerCase( )
    public String toUpperCase( )
    public String toUpperCase( )
    public String toString( )
    public char[ ] toCharArray( )
    …
}
```



Java functions

- A group of statements
  - To perform a task
  - Possibly return a value
- Unlike C, functions are part of a class in Java

- Syntax

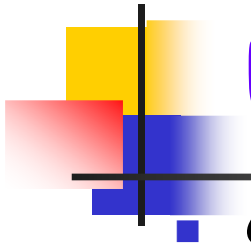
```
<return_type> fn_name (arguments)
    {
        // function body
    }
```

Java functions … example

```
import java.io.*;

public class anyClass                          // Define a class first
{
   int square (int x)                          // function to compute square
     {
        return (x * x);
     }

   public static void main(String args[ ])  //Starting point of the program
     {
        int i = 5;
        anyClass ac = new anyClass();          // Create an object first
        int i_sq = ac.square (5);              // Call its function
        System.out.println ("Square of 5 is: " + i_sq);
     }
}
```



Useful Java functions

- clone ( )
  - Create a copy of an existing object
- equals ( )
  - Checks if two objects are the same
  - This is not quite the same as == operator
- finalize ( )
  - Called to clean up object's resources.
- getClass ( )
  - Returns a class object
- hashCode ( )
  - Returns object's memory address in hexadecimal
- toString ( )
  - Returns a string representation of the object.