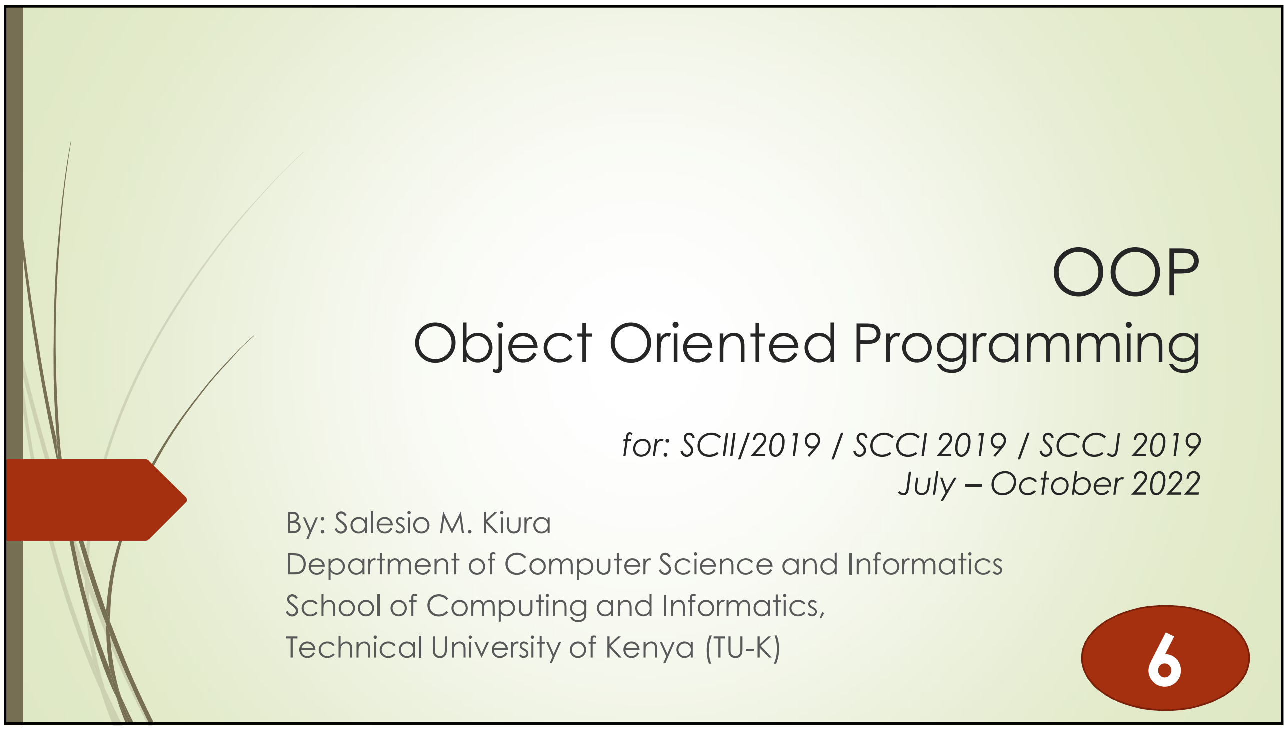


OOP
Object Oriented Programming

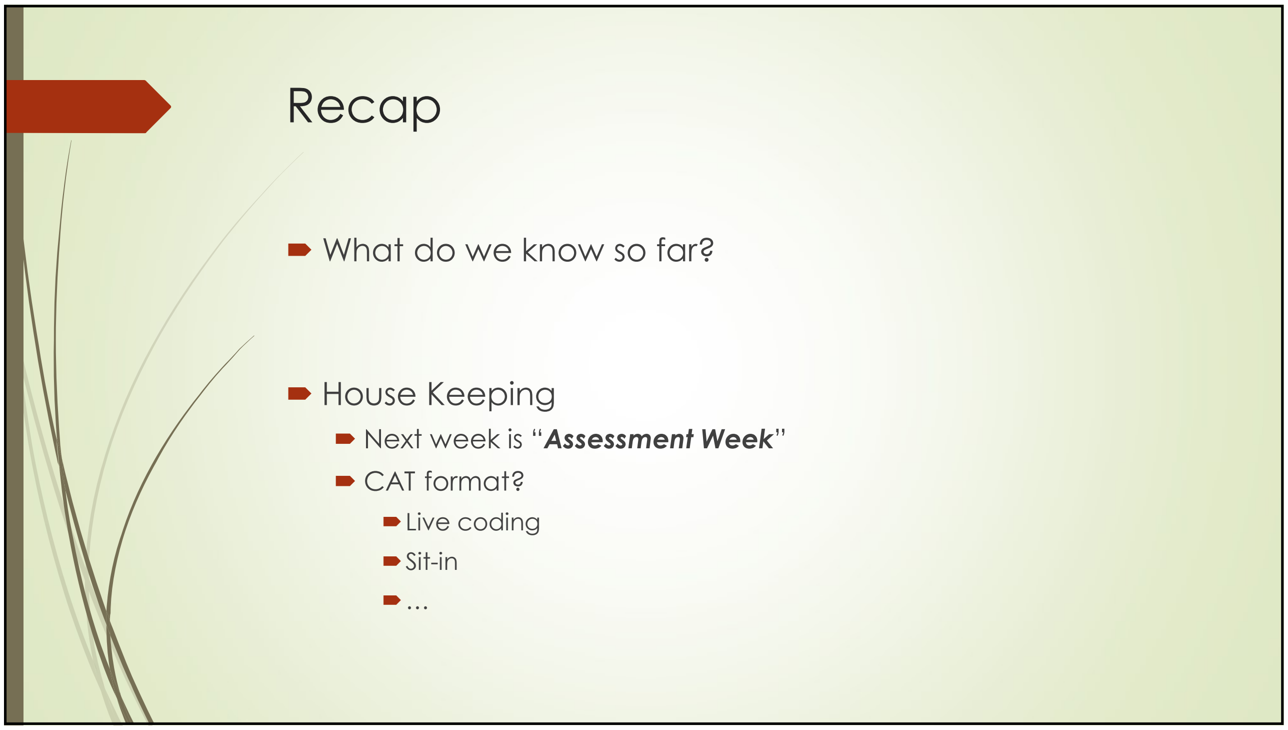
for: SCII/2019 / SCCI 2019 / SCCJ 2019
July – October 2022

By: Salesio M. Kiura
Department of Computer Science and Informatics
School of Computing and Informatics,
Technical University of Kenya (TU-K)

6

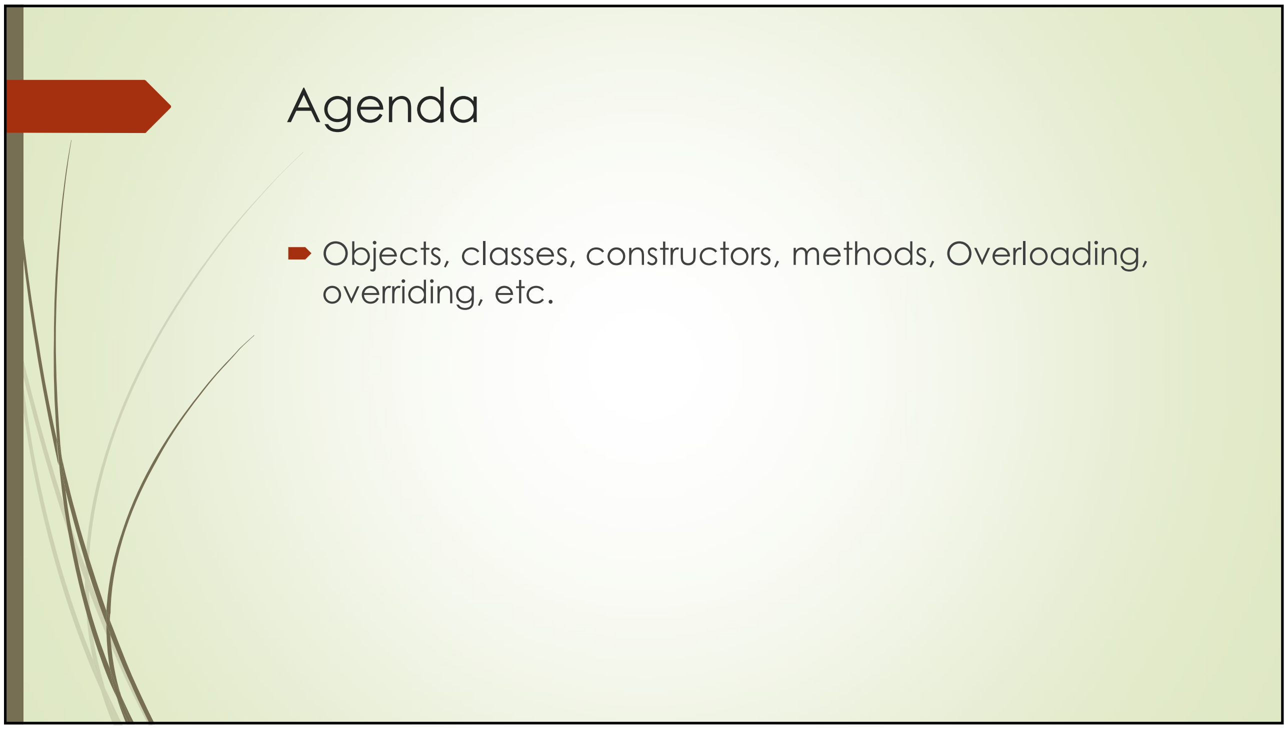
Session 4, 5: Contd…
OOP programs parts, mini project description, basic concepts implementation

28th July 2022



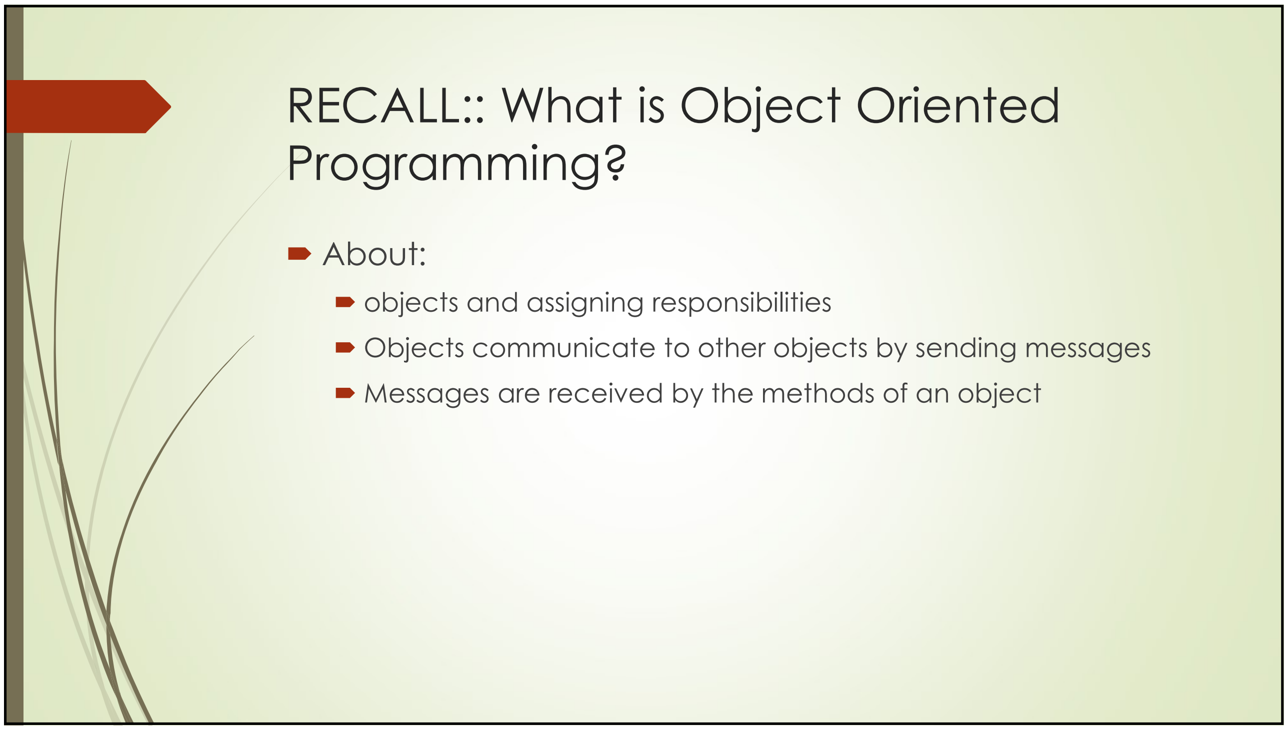
Recap

- What do we know so far?
- House Keeping
  - Next week is "***Assessment Week***"
  - CAT format?
    - Live coding
    - Sit-in
    - …



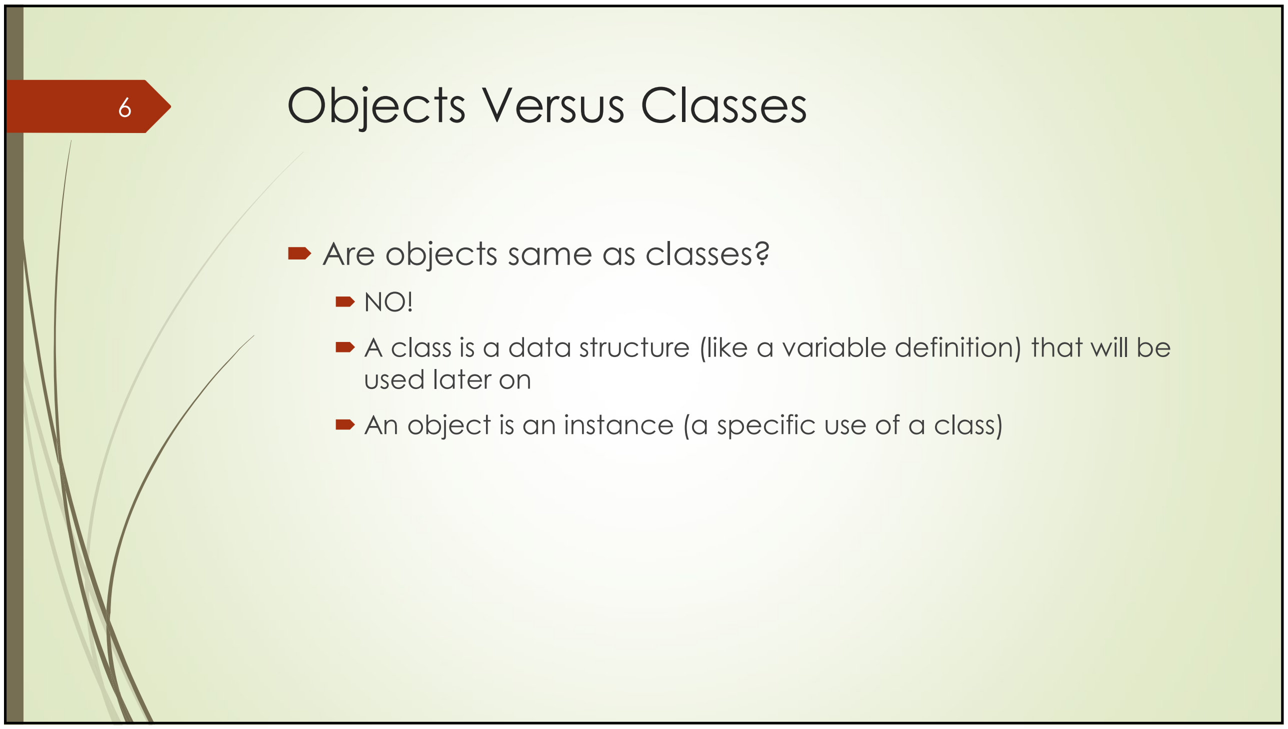
Agenda

- Objects, classes, constructors, methods, Overloading, overriding, etc.



RECALL:: What is Object Oriented Programming?

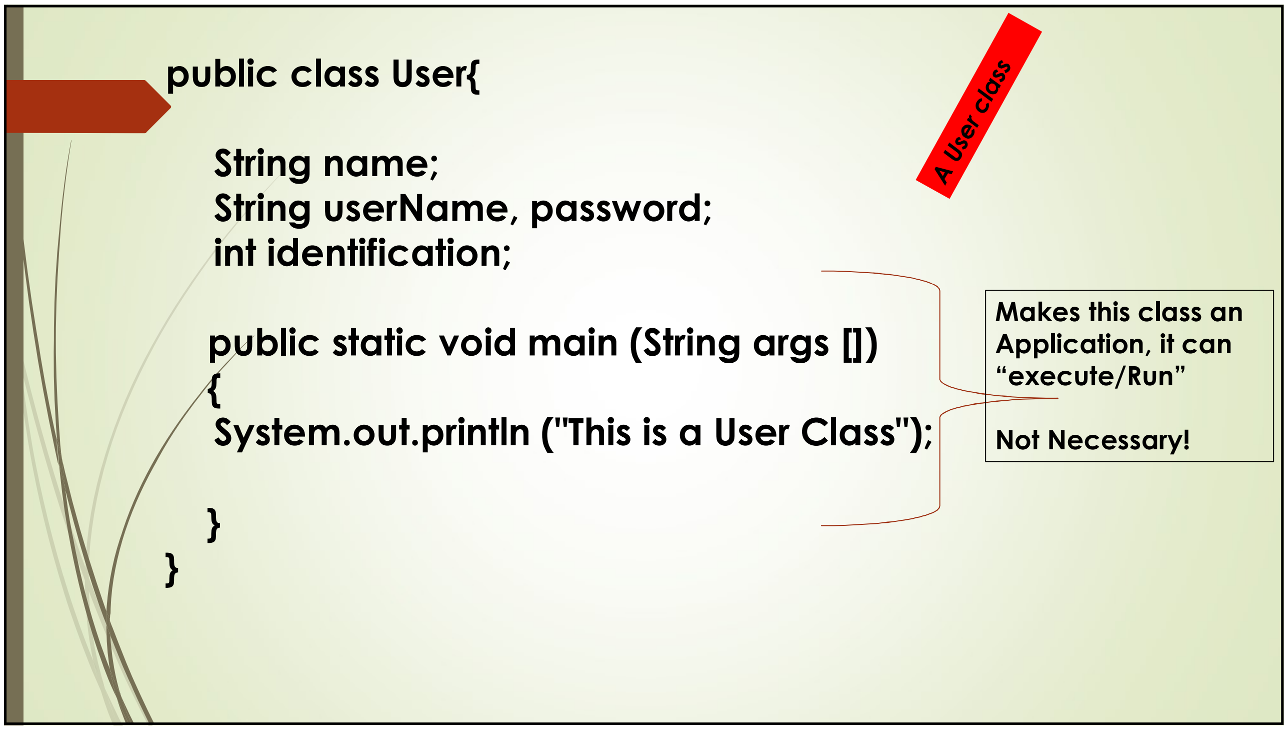
- About:
  - objects and assigning responsibilities
  - Objects communicate to other objects by sending messages
  - Messages are received by the methods of an object



6

Objects Versus Classes

- Are objects same as classes?
  - NO!
  - A class is a data structure (like a variable definition) that will be used later on
  - An object is an instance (a specific use of a class)

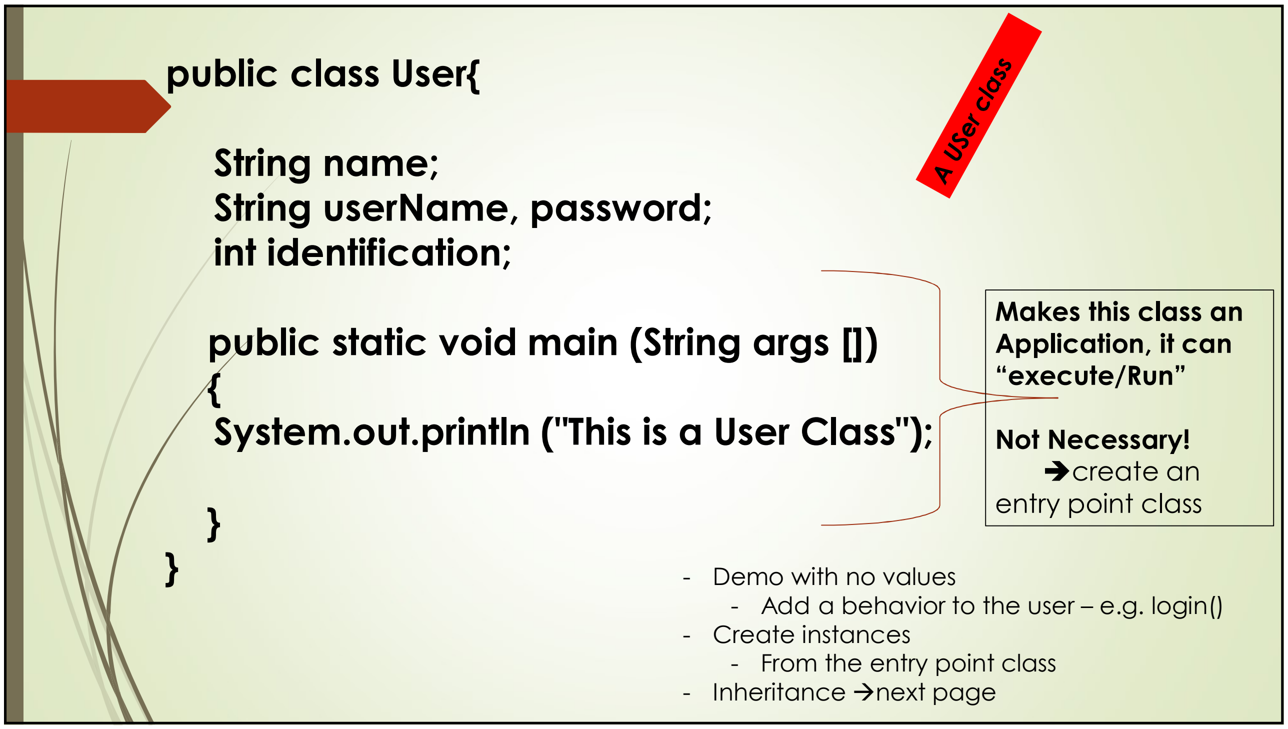


```
public class User{

    String name;
    String userName, password;
    int identification;

    public static void main (String args [])
    {
    System.out.println ("This is a User Class");

    }
}
```

A User class

Makes this class an Application, it can "execute/Run"

Not Necessary!



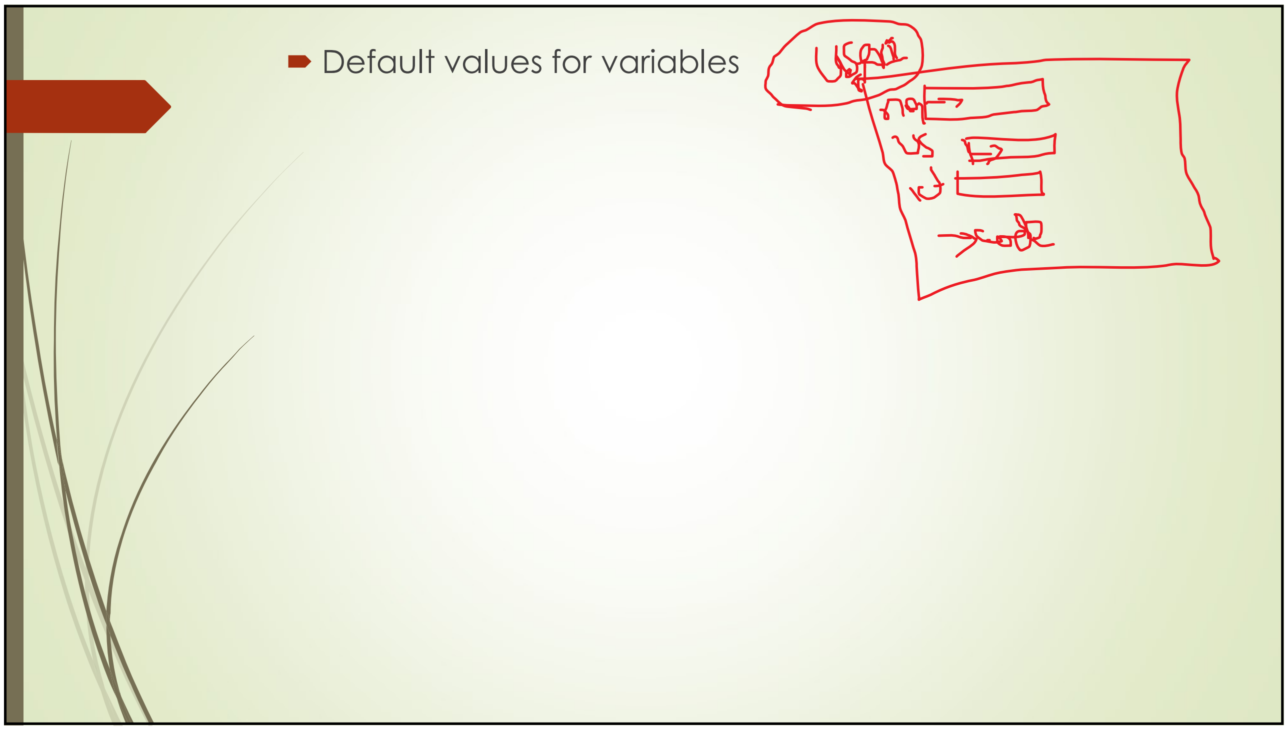
```
public class User{

    String name;
    String userName, password;
    int identification;

    public static void main (String args [])
    {
    System.out.println ("This is a User Class");

    }
}
```

A USer class

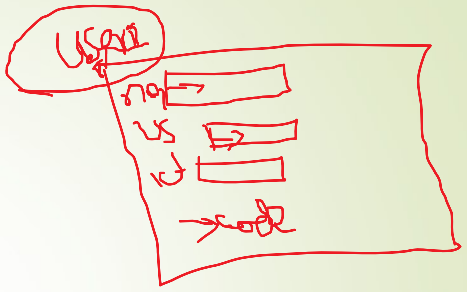
Makes this class an Application, it can "execute/Run"

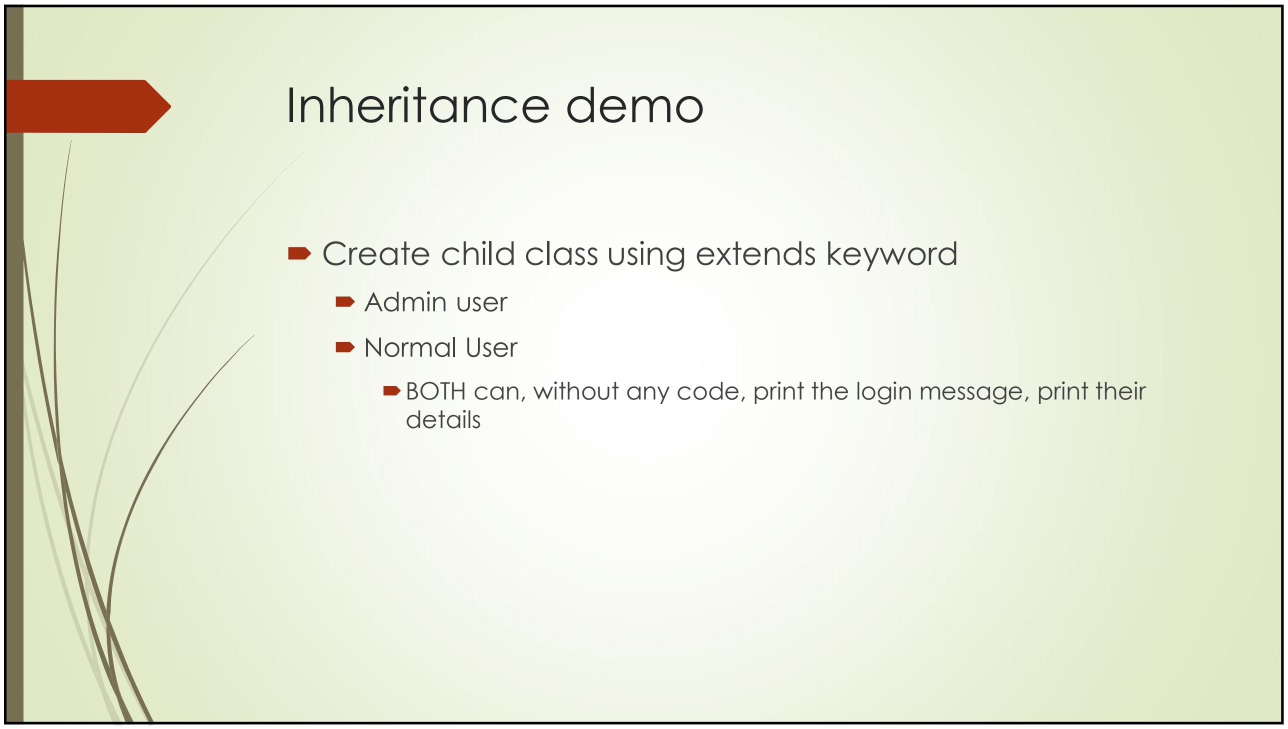
Not Necessary!
→create an entry point class

- Demo with no values
  - Add a behavior to the user – e.g. login()
- Create instances
  - From the entry point class
- Inheritance →next page



- Default values for variables





Inheritance demo

- Create child class using extends keyword
  - Admin user
  - Normal User
    - BOTH can, without any code, print the login message, print their details

Add functionalities to the classes

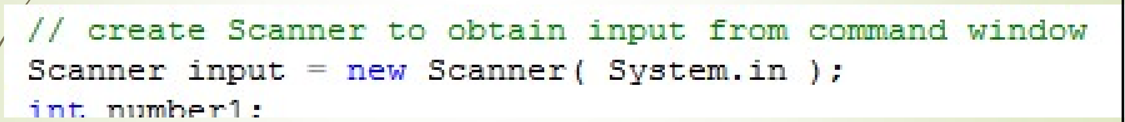
User Class **SystemAdmin Class** **NormalUser Class**

- We revisit the **flow chart** /Algorithm of the APP
  - Get user input :: select what a user wants to do
  - Register:
    - Prompt for details
    - …
  - Login
    - Prompt for a username and password

- What are objects?
  - an object represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain.
  - An "object" is anything to which a concept applies

- We have identified and implemented classes that represent Objects in the scenario/group project assignment earlier

- How are objects created?
  - A class ***constructor*** is used to create instances of a class by using the ***new*** keyword
  - Recall



```
// create Scanner to obtain input from command window
Scanner input = new Scanner( System.in );
int number1;
```

- Class constructor
  - **Constructor name is class name**. A constructor must have the *same name as the class its in*
  - **Default constructor**. If you don't define a constructor for a class, a *default parameterless constructor* is automatically created by the compiler. The default constructor calls the default parent constructor (super() from the Object class) and initializes all instance variables to default value (zero for numeric types, null for object references, and false for booleans).

- **Default constructor is created only if there are no constructors**. If you define *any* constructor for your class, no default constructor is automatically created.
- **Differences between methods and constructors**. There is *no return type* given in a constructor signature (header). The value is this object itself so there is no need to indicate a return value.
  - There is *no return statement* in the body of the constructor.
  - The *first line* of a constructor must either be a call on another constructor in the same class (using this), or a call on the superclass constructor (using super). If the first line is neither of these, the compiler automatically inserts a call to the parameterless super class constructor.

Example – user class Constructor

- User class constructor
  - Register() method in user class :: creates an instance
- Admin User Constructor
  - Updating a field that only Admin has
- Normal User constructor

Objects interaction - Exercise

- See next slides

```
package tuk;
/**
 * @author Salesio
 */
public class TUKGateSystem{
   /**
    This class is the Application entry
      point*/
   public static void main(String[] args) {
      // TODO code application logic here
      System.out.println("I am the main
method");
      Person person1 = new Person(2,"Salesio");
      System.out.println(person1.password);
   }
}
```

```
package tuk;
// @author Salesio
public class Person {
   int id;
   String name;
   String password;
   public Person(int anId, String aName){
      id=anId;
      name = aName;
    }
}
```

What is the Output?

What is the Output, now?

```
package tuk;
/**
 * @author Salesio
 */
public class TUKGateSystem{
  /**
   This class is the Application entry
     point*/
  public static void main(String[] args) {
     // TODO code application logic here
     System.out.println("I am the main method");
     Person person1 = new Person(2,"Salesio");
     System.out.println(person1.password);
  }
}
```

```
package tuk;
// @author Salesio
public class Person {
   int id;
   String name;
   String password;
   public Person(int anId, String aName){
      id=anId;
      name = aName;
password = Security().createPassword(id,
name);
     } }
```

```
package tuk;
/** @author Salesio */
class Security {
    String createPassword(int id,
String name) {
        String pass =
id+name.substring(1);
        return pass;
    }
}
```

- Why would having two or more classes above make sense?
  - Easy maintenance!
  - E.g. assume that the rules of generating usernames change. We only have to change and compile the class that generates Ids/Usernames

Overriding

- ?

Overloading

- Thanks

Next

- Objects and Classes (personal work implementation
- Methods, Java parameter passing
- mini project (initial Assessment)