IS 2104 - Rapid Application Development

Object Oriented Concepts

M.V.P. Thilini Lakshika
University of Colombo School of Computing
tlv@ucsc.cmb.ac.lk

Lesson Outline

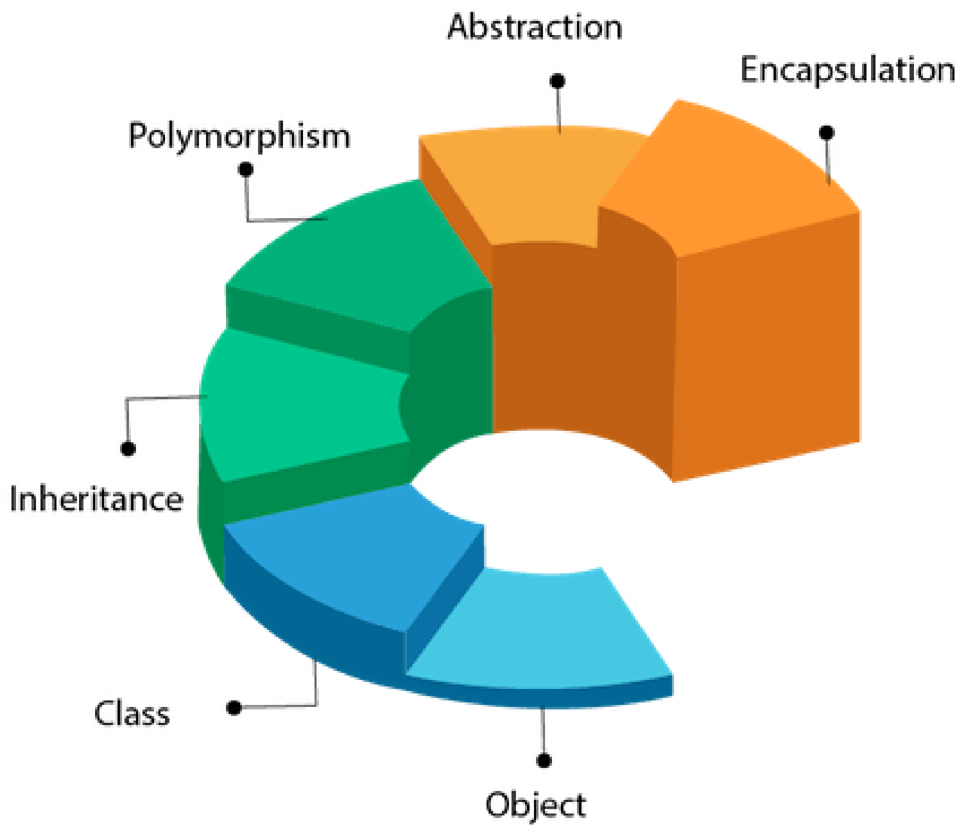
- Object Oriented Programming
- Java Naming conventions
- Object
- Class
- Method
- Comparison of OOPS with other programming styles
- Advantages of OOP

Object Oriented Programming (OOP)

- Object-Oriented Programming (OOP) is a methodology or paradigm to design a program using classes and objects.
- OOP works on the principle that objects are the most important part of your program.
- Object means a real-world entity such as a student, chair, car, computer.
- Manipulating these objects to get results is the goal of OOP.
- OOP allows user to create the objects that they want and then create methods to handle those objects.

Object Oriented Programming (OOP)

- OOP simplifies software development and maintenance by providing some concepts:
  - Object
  - Class
  - Inheritance
  - Polymorphism
  - Abstraction
  - Encapsulation



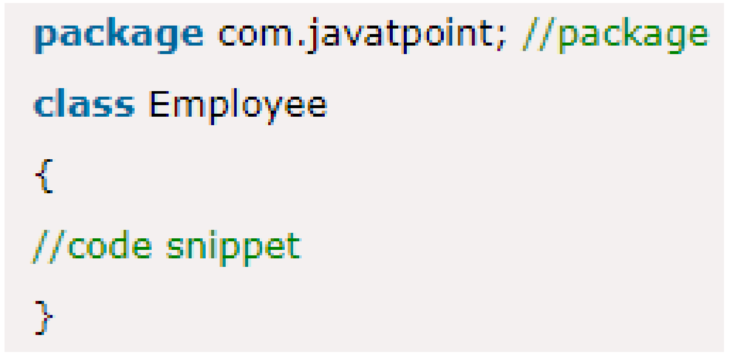
Java Naming conventions

- Follow Java naming conventions when you decide names for your identifiers such as class, package, variable, constant, method, etc.
- This makes your code easier to read for yourself and other programmers.
- These conventions are suggested by several Java communities such as Sun Microsystems and Netscape.
- Some conventions must be followed by every identifier:
  - The name must not contain any white spaces.
  - The name should not start with special characters like &, $, _
  - Follow camel-case syntax for naming. (Ex: actionPerformed(), getArea(), firstName, maxSpeed)
  - Use appropriate words, instead of acronyms.

Java Naming conventions

Package

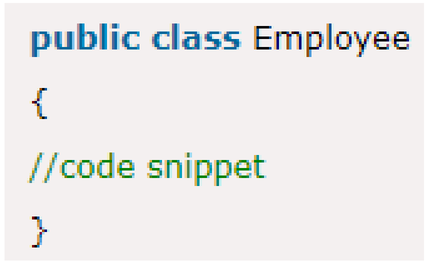
- It should be a lowercase letter such as java, lang.
- If the name contains multiple words, it should be separated by dots (.)
  Ex: java.util, java.lang.



```
package com.javatpoint; //package
class Employee
{
//code snippet
}
```

Class

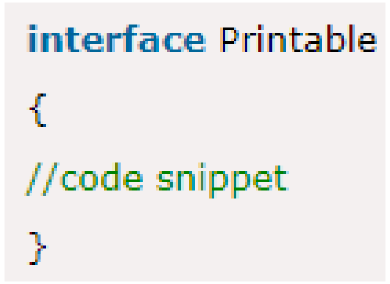
- It should start with the uppercase letter.
- It should be a noun.



```
public class Employee
{
//code snippet
}
```

Interface

- It should start with the uppercase letter.
- It should be an adjective.



```
interface Printable
{
//code snippet
}
```

Java Naming conventions

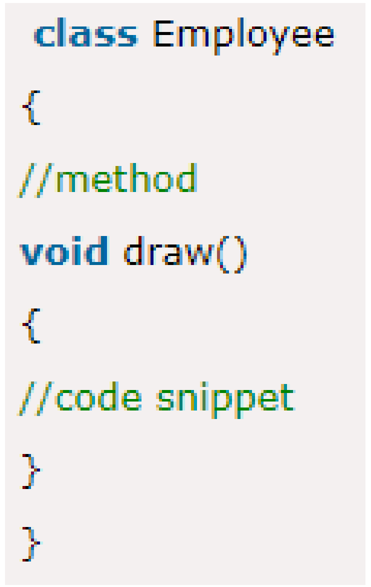
Method

- It should start with lowercase letter.
- It should be a verb such as main(), print(), println().
- If the name contains multiple words, start it with a lowercase letter followed by an uppercase letter such as actionPerformed().

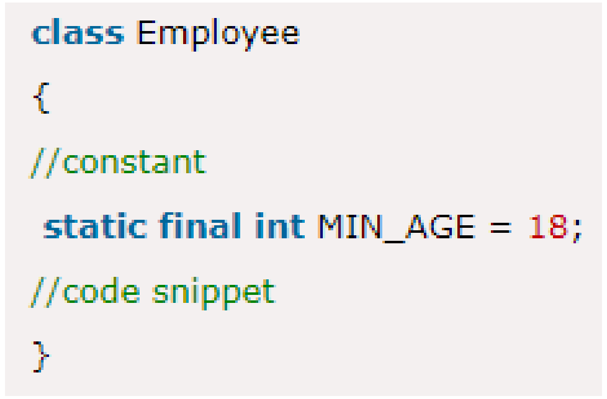
Constant

- It should be in uppercase letters such as RED, YELLOW.
- If the name contains multiple words, it should be separated by an underscore(_)

  Ex: MAX_PRIORITY
- It may contain digits but not as the first letter.



```
class Employee
{
//method
void draw()
{
//code snippet
}
}
```



```
class Employee
{
//constant
 static final int MIN_AGE = 18;
//code snippet
}
```

Object

- An object represents an entity in the real world that can be distinctly identified.
- It can be physical or logical.

  Ex: student, car, table, car, pen, circle, loan
- An object has a unique identity, state and behavior.
- The state (properties or attributes) of an object is represented by data fields with their current values.

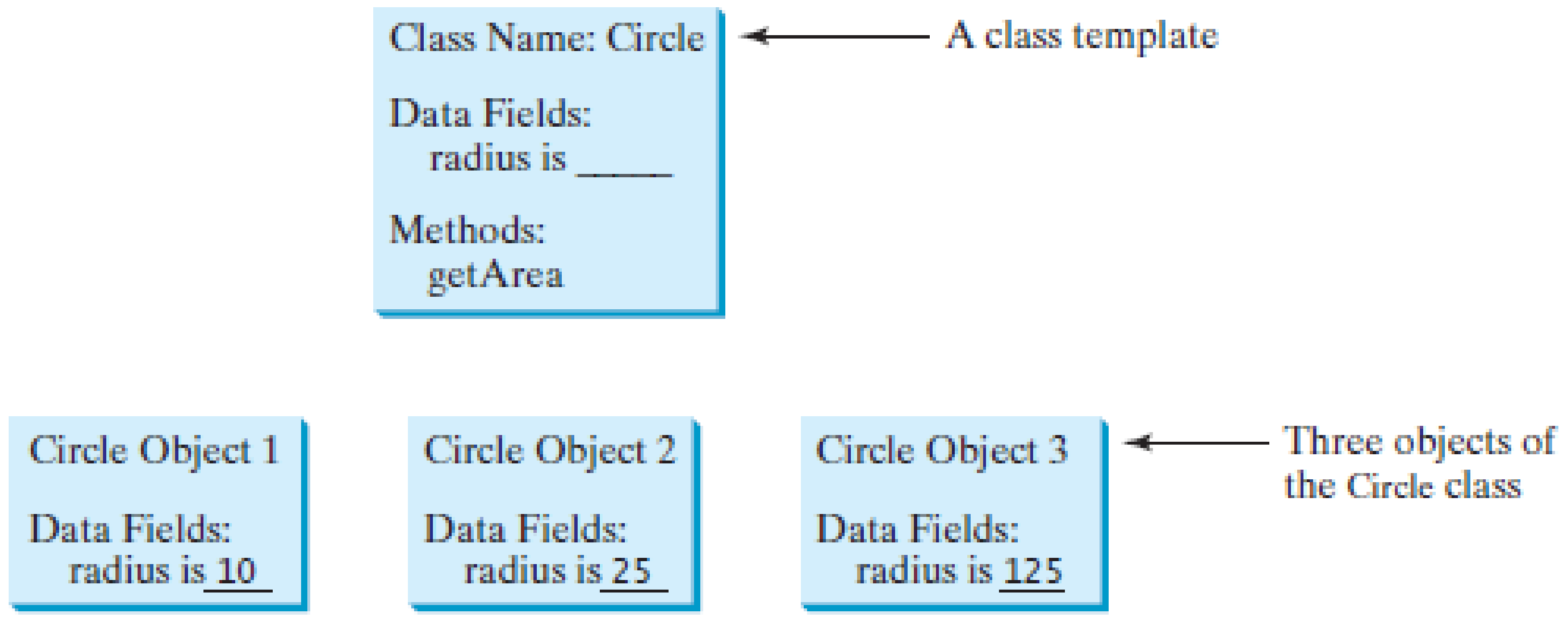
  Ex: Object – Student, Properties – name, age, school, grade

  Object – Car, Properties – color, fuel type, manufactured country
- The behavior (actions) of an object is represented by methods.

  Ex: Object – Car Methods - getSpeed(), fullThrottle()
- Object is performing an action when invoking a method on an object.

Class

- A class is a template or blueprint that defines the form (data fields and methods) of an object.
- A class is essentially a set of plans that specify how to build an object.
- Objects are instances of a class. You can create multiple instances of a class.
- Java uses a class specification to construct objects.



Class

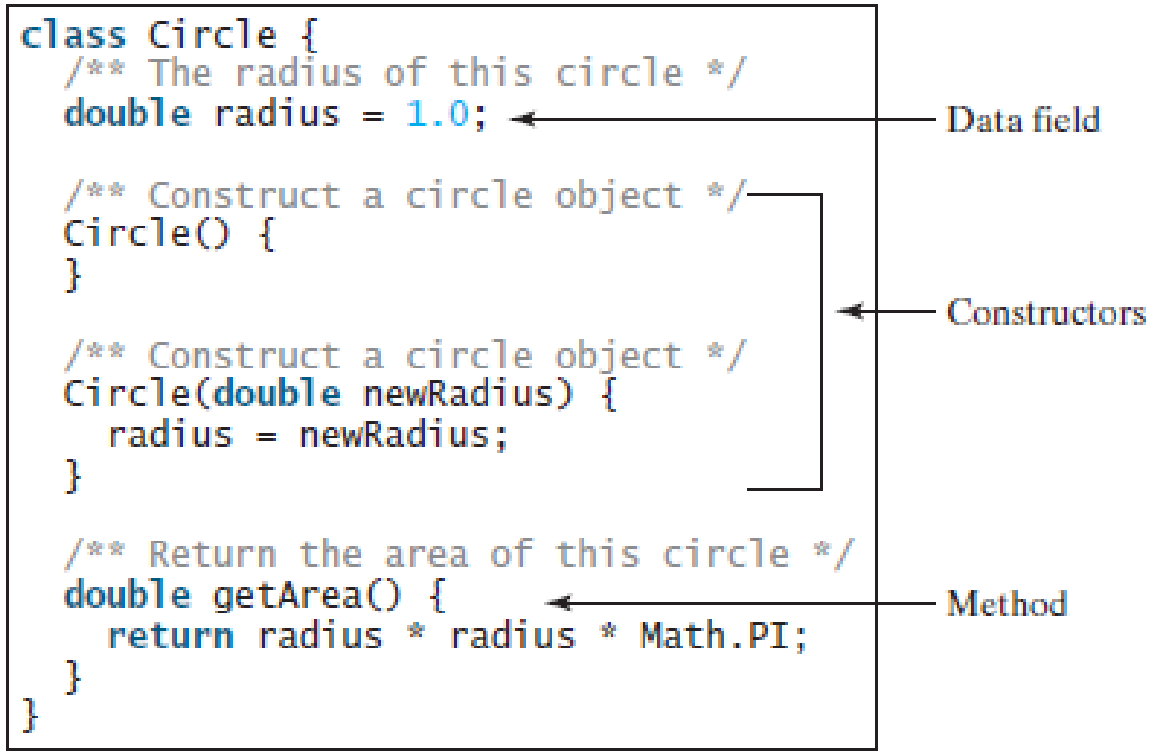
- A class is created by using the keyword **class**.
- A Java class uses variables to define data fields and methods to define actions.
- The data members are also referred to as instance variables.
- A class provides a special type of methods known as constructors which are invoked to create a new object.
- A constructor is designed to perform initializing actions such as initializing the data fields of objects.

```
class classname {
// declare instance variables
type var1;
type var2;

// declare methods
type method1(parameters) {
   // body of method
          }

type method2(parameters) {
   // body of method
          }
}
```

General form of a **class** definition.

Class

- The **Circle** class does not have a **main** method and therefore cannot be run.
- It is merely a definition for circle objects.



```
class Circle {
  /** The radius of this circle */
  double radius = 1.0;

  /** Construct a circle object */
  Circle() {
  }

  /** Construct a circle object */
  Circle(double newRadius) {
    radius = newRadius;
  }

  /** Return the area of this circle */
  double getArea() {
    return radius * radius * Math.PI;
  }
}
```

Defining the **class** for circle objects.

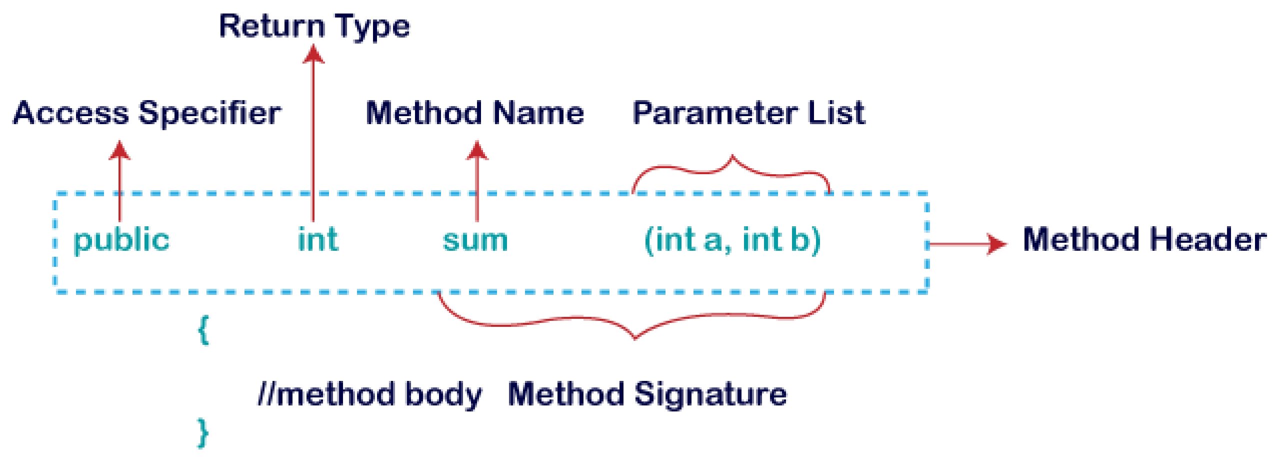
Class

```
public class TestCircle {
public static void main(String[] args) {
   // Create a circle with radius 1.0
   Circle circle1 = new Circle();
   System.out.println("The area of the circle of radius " + circle1.radius + " is " + circle1.getArea());
   // Create a circle with radius 25
   Circle circle2 = new Circle (25);
   System.out.println("The area of the circle of radius "  + circle2.radius + " is " + circle2.getArea());
   // Create a circle with radius 125
   Circle circle3 = new Circle (125);
   System.out.println("The area of the circle of radius " + circle3.radius + " is " + circle3.getArea());
   // Modify circle radius
   circle2.radius = 100;
   System.out.println("The area of the circle of radius " + circle2.radius + " is " + circle2.getArea());
               }
}
```

Methods

- A method is a block of code or collection of statements to perform a certain task.
- You can write a method once and use it many times.
- The method is executed only when we call or invoke it.
- Methods are used to achieve the reusability of code.
- It also provides the easy modification (adding or removing a chunk of code) and readability of code.
- The most important method in Java is the main() method.

Methods



Method Declaration

- Access specifier or modifier specifies the visibility of the method. Java provides 4 types of access specifier as public, private, protected and default.
- Return Type is a data type that the method returns. If the method does not return anything, use **void** keyword.
- Method Name must be corresponding to the functionality of the method.
- Parameter List is a comma separated list of parameters. It contains the data type and variable name.
- Method Body contains all the actions to be performed.
- Method Signature includes the method name and parameter list.

Comparison of OOPS with other programming styles

- Programming languages can be classified into 3 primary types.
  - Unstructured Programming Languages: The most earliest and primitive of all programming languages. Having sequential flow of control and code is repeated through out the program.
  - Structured Programming Languages: Has non-sequential flow of control. Use of functions allows for re-use of code.
  - Object Oriented Programming: Combines Data & Action together. Greater level of reuse.

Comparison of OOPS with other programming styles

- Suppose you want to create a Banking Software with functions like Deposit, Withdraw and Show Balance.

Example with Unstructured Programming Languages

- This is a very elementary code of banking application with two variables.

```
int account_number=20;
int account_balance=100;
```

- Suppose deposit of 100 dollars is made.

```
account_balance=account_balance+100;
```

- Next you need to display account balance.

```
printf(“Account Number=%d,account_number);
printf(“Account Balance=%d,account_balance);
```

Comparison of OOPS with other programming styles

- Now the amount of 50 dollars is withdrawn.

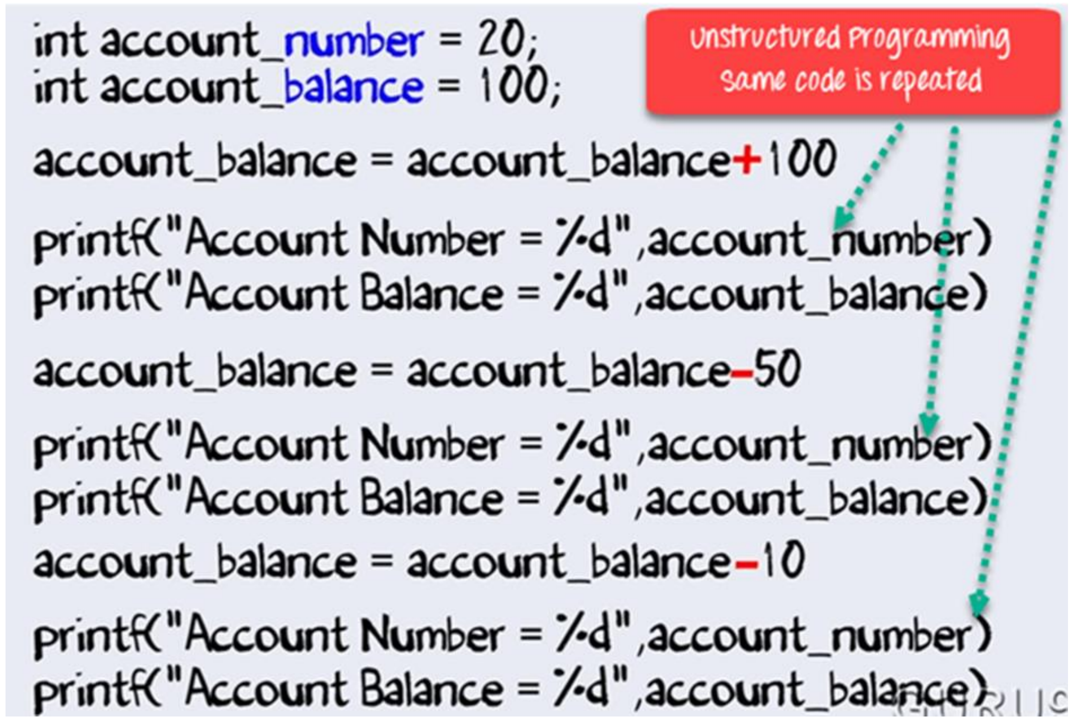
  account_balance=account_blance-50;

- Again, you need to display the account balance.

  printf(“Account Number=%d,account_number);

  printf(“Account Balance=%d,account_balance);

- For any further deposit or withdrawal operation, you will repeat the same lines again and again.

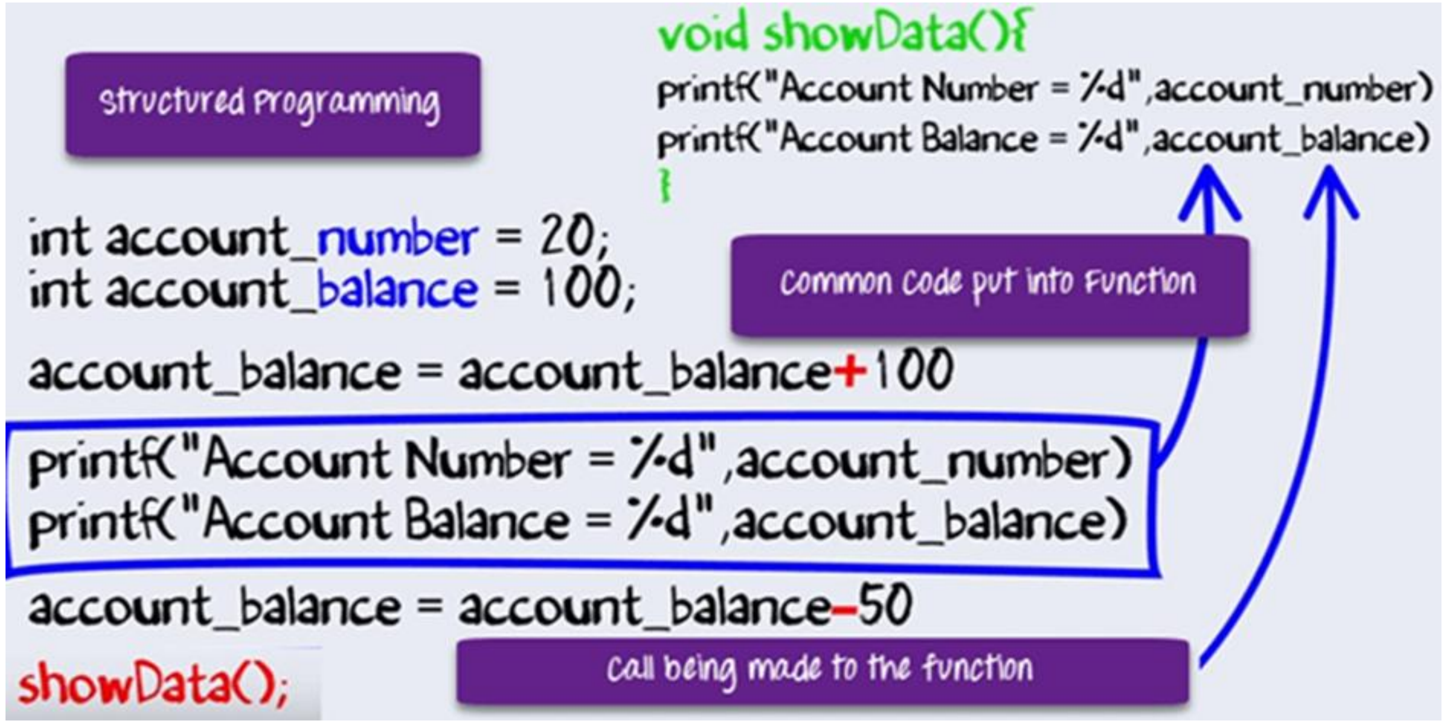


```
int account_number = 20;
int account_balance = 100;

account_balance = account_balance+100

printf("Account Number = %d",account_number)
printf("Account Balance = %d",account_balance)

account_balance = account_balance-50

printf("Account Number = %d",account_number)
printf("Account Balance = %d",account_balance)
account_balance = account_balance-10

printf("Account Number = %d",account_number)
printf("Account Balance = %d",account_balance)
```

Comparison of OOPS with other programming styles

Example with Structured Programming Languages

- With the arrival of structured programming, repeated lines on the code were put into structures such as functions or methods.
- Whenever needed, a simple call to the function is made.


Structured Programming
Common code put into Function
Call being made to the function

```
void showData(){
printf("Account Number = %d",account_number)
printf("Account Balance = %d",account_balance)
}
```

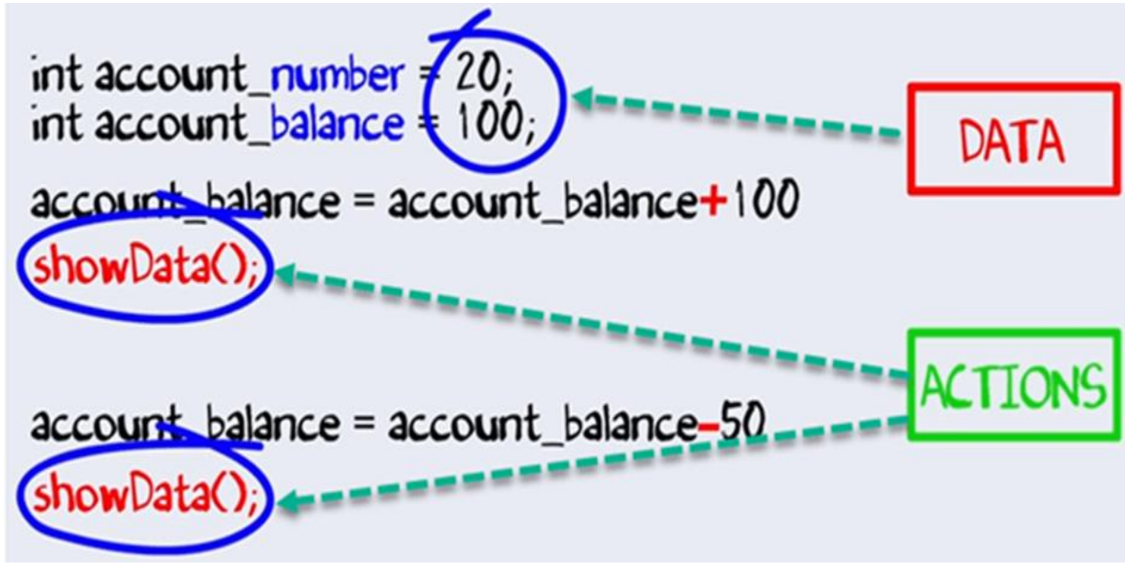
```
int account_number = 20;
int account_balance = 100;
account_balance = account_balance+100
printf("Account Number = %d",account_number)
printf("Account Balance = %d",account_balance)
account_balance = account_balance-50
showData();
```

Comparison of OOPS with other programming styles

Example with Object-Oriented Programming

- Experts in Software Programming thought of combining the Data and Operations.
- The same code in OOP will have same data and some action performed on that data.

```
Class Account{
    int account_number;
    int account_balance;
public void showData(){
    system.out.println("Account Number"+account_number);
    system.outprintln("Account Balance"+ account_balance);
    }
}
```



```
int account_number = 20;
int account_balance = 100;
account_balance = account_balance+100
showData();

account_balance = account_balance-50
showData();
```

Advantages of OOP

- OOP is easy to understand and makes development and maintenance easier, whereas, in unstructured programming language, it is not easy to manage if code grows as project size increases.
- OOPs provides data hiding, whereas, in structured programming languages, global data can be accessed from anywhere.
- OOPs provides the ability to simulate real-world event much more effectively.
- Objects created for Object-Oriented Programs can be reused in other programs. Thus it saves significant development cost.
- OOP offers a clear modular structure for programs because every object exists independently.

END