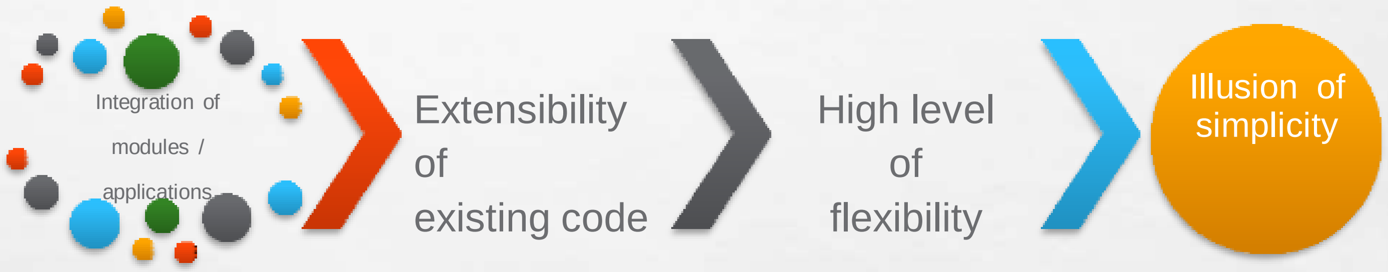
OBJECT ORIENTED PROGRAMMING CONCEPTS IN JAVA

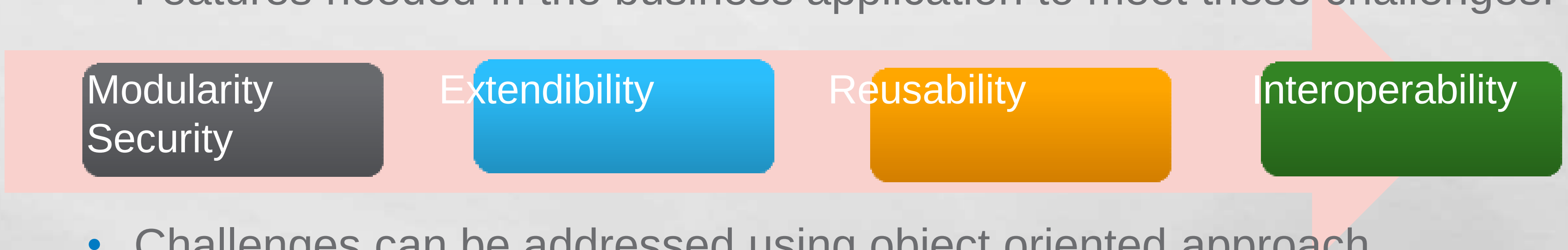
- OBJECT-ORIENTED PROGRAMMING IS THE PROBLEM-SOLVING APPROACH AND USED WHERE COMPUTATION IS DONE BY USING OBJECTS

NEED FOR OBJECT ORIENTED APPROACH

- CHALLENGES IN DEVELOPING A BUSINESS APPLICATION


Integration of modules / applications
Extensibility of existing code
High level of flexibility
Illusion of simplicity

- If these challenges are not addressed it may lead to **Software Crisis**
- Features needed in the business application to meet these challenges:

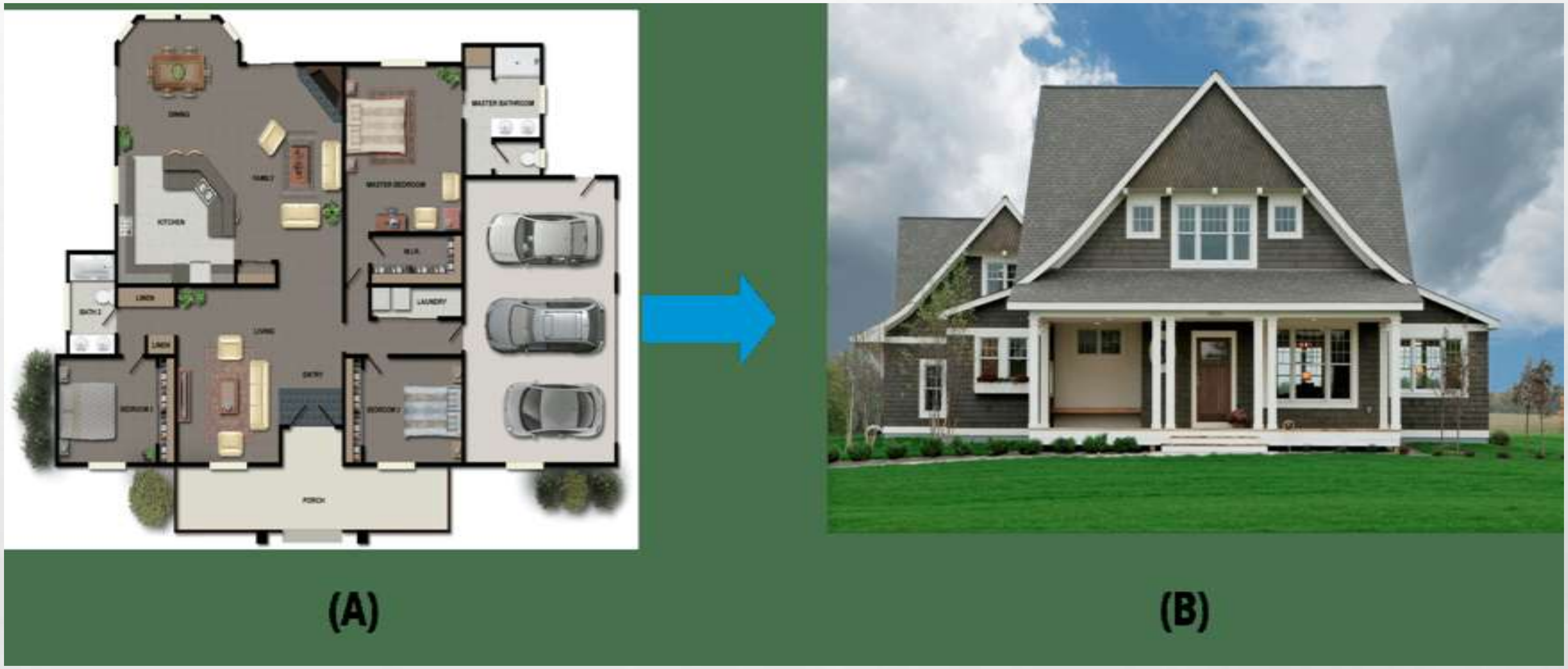


- Challenges can be addressed using object oriented approach

OOP TERMINOLOGIES

Object	Real world entities, which has two characteristics namely, state (attributes) and behavior (method). It is an active entity.
Class	A Class is a "blueprint" for creating objects.
Method	object's data is accessed, modified, or processed
Abstraction	Hides all but the relevant data about an object so as to reduce complexity and increase efficiency. Focus on what the object does instead of how it does
Encapsulation	This wraps code and data into a single unit. implementation of abstraction.
Inheritance	Inheritance allows us to define a class that inherits all the methods and properties from another class
Polymorphism	Polymorphism means that different types respond to the same function

Classes & Objects



- ***Users of the retail application – Billing staff, Admin, Retail outlet manager***
- ***Each user needs to know some details and need not know other details***

Who are the users of the retail application?

What are the things each user must know to perform their activities ?

Billing staff (Billing of customers)

Admin (Registration of customers)

Retail Outlet Manager (Registration of users)

ABSTRACTION : Process of identifying the essential details to be known and ignoring the non-essential details from the perspective of the user of the system

ENCAPSULATION

- Swipe machine in a retail store
  - Used by billing staff to key the amount
  - Used by admin to record payment

How is a swipe machine used for payment of bill in a retail store?

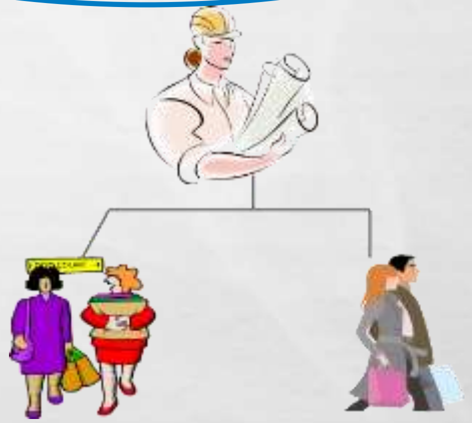
ENCAPSULATION : A mechanism of hiding the members from the external world. Swipe machine has all the control to perform the operations.

INHERITANCE

Customers are of two kinds

- Regular
- Privileged

What are the two different types of customers you can see in the retail application?



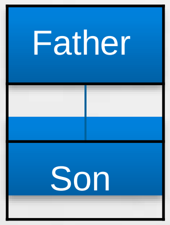
All customers have Customer Id, Name, Telephone Number and Address

The regular customer in addition is given discounts

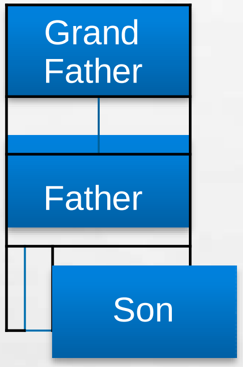
All customers have some generic features. The different kinds of customers have all generic features in addition to some specific features

INHERITANCE : Is a mechanism which allows to define generalized characteristics and behavior and also create specialized ones. The specialized ones automatically tend to inherit all the properties of the generic ones

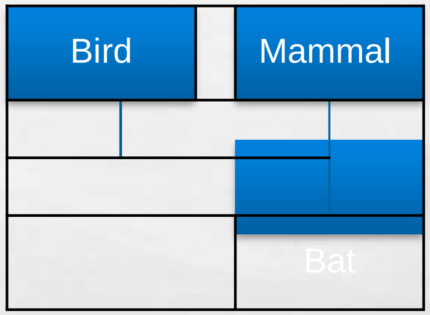
TYPES OF INHERITANCE



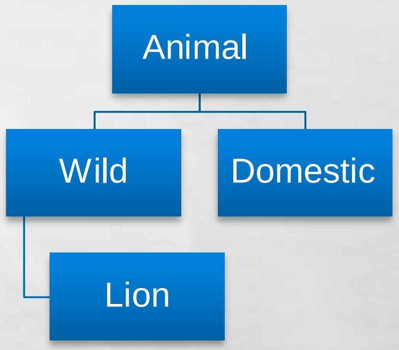
Single Inheritance



Multi-level Inheritance



Multiple Inheritance



Hierarchy Inheritance

POLYMORPHISM

What do you observe in this retail store scenario?

- Payment of bill - Two modes
  - Cash (Calculation includes VAT)

Total Amount = Purchase amount + VAT

  - Credit card(Calculation includes processing charge and VAT)

Total Amount = Purchase amount + VAT
+ Processing charge

POLYMORPHISM: Refers to the ability of an object/operation to behave differently in different situations

OBJECT ORIENTED APPROACH – BENEFITS

- Leads to development of smaller but stable subsystems
- The subsystems are resilient to change
- Reduces the risk factor in building large systems as they are built incrementally from subsystems which are stable

Hence Object Orientation is suitable for developing extremely complex business systems

ACCESS SPECIFIES

- Used to expose or hide the attribute and behavior of a class
- Used to specify access permissions on a member variable/function/method

Naming	Type	UML Notation	Meaning
name	Public	+	These attributes can be freely used inside or outside of a class definition. A declaration that is accessible to all classes
_name	Protected	#	Protected attributes are accessible only within the class in which it is declared and to its inherited sub-classes.
__name	Private	-	This kind of attribute is **inaccessible and invisible**. It's neither possible to read nor write to those attributes, except inside of the class definition itself. A declaration that is accessible only to the class in which it is declared

Activity

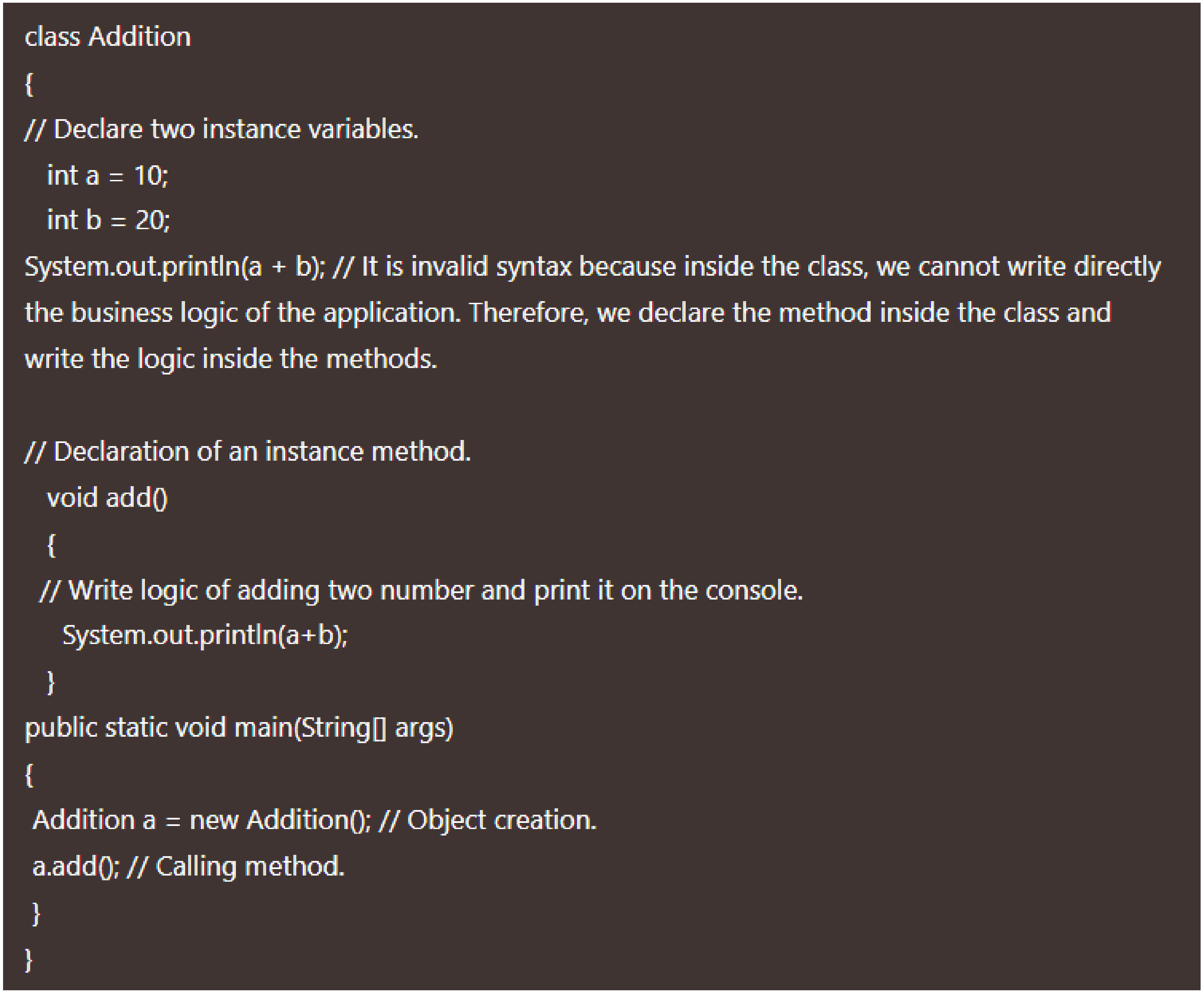
- Program print “Hello World”

```
public class Helloworld {
   public static void main(String[] args) {
      System.out.println("Hello, World");
   }
}
```

Test Data:NA
Output:Hello, World

```
• public class Main {
•   void myMethod() {
•     System.out.println("I just got executed!");
•   }

•   public static void main(String[] args) {
•     myMethod();
•   }
• }
```

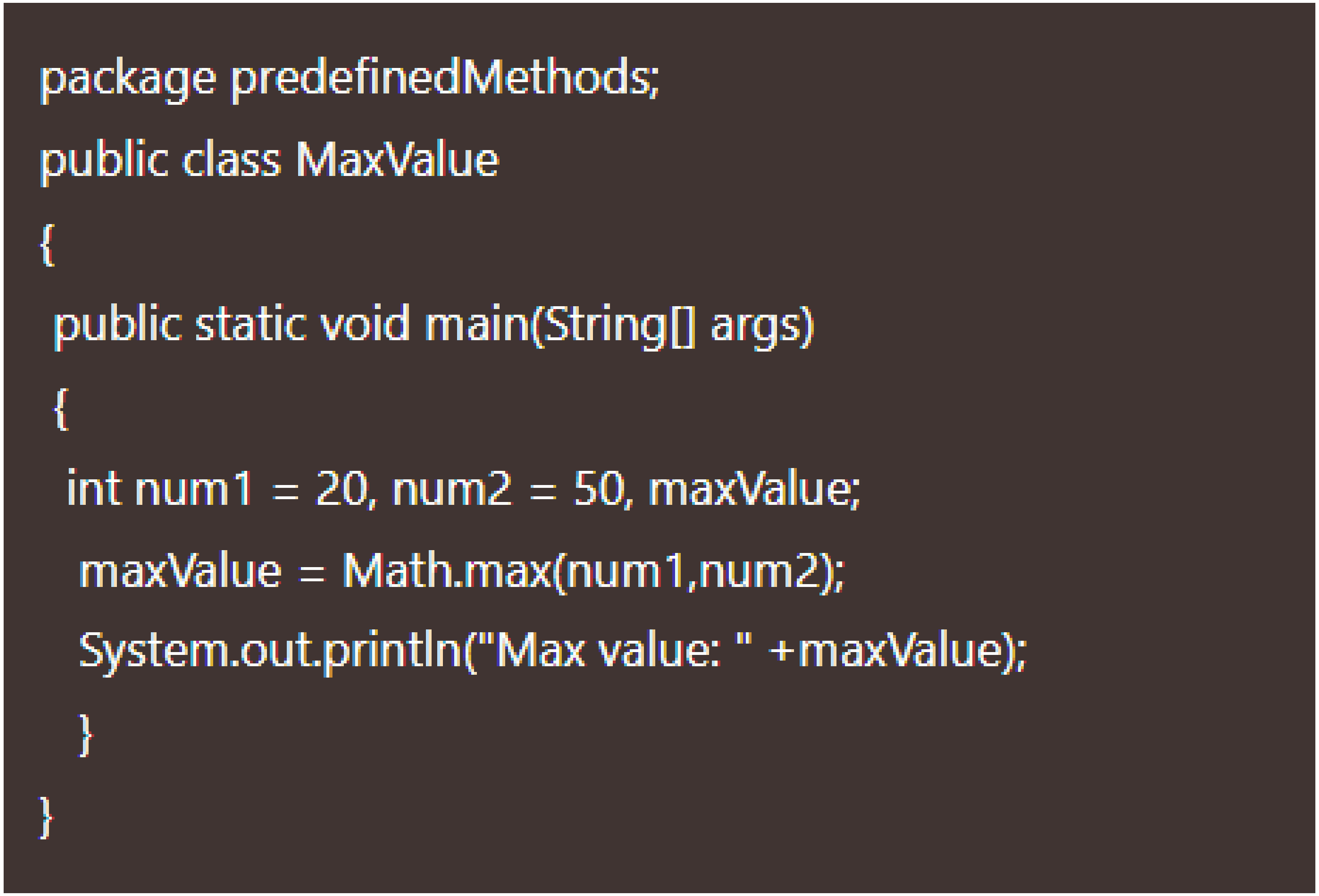


```
class Addition
{
// Declare two instance variables.
   int a = 10;
   int b = 20;
System.out.println(a + b); // It is invalid syntax because inside the class, we cannot write directly
the business logic of the application. Therefore, we declare the method inside the class and
write the logic inside the methods.

// Declaration of an instance method.
   void add()
   {
  // Write logic of adding two number and print it on the console.
     System.out.println(a+b);
   }
public static void main(String[] args)
{
 Addition a = new Addition(); // Object creation.
 a.add(); // Calling method.
 }
}
```

Write a Program to print the area of triangle. Save it with name Area.java in your folder.

```
class
Area {
 public static void main(String args[]) {
 int height =10,base=6;
float area=0.5F*base* height;
System.out.println(“area of triangle = ”+area);
}
}
```



```
package predefinedMethods;
public class MaxValue
{
 public static void main(String[] args)
 {
  int num1 = 20, num2 = 50, maxValue;
   maxValue = Math.max(num1,num2);
   System.out.println("Max value: " +maxValue);
   }
}
```