پوهنتون کاردان
KARDAN UNIVERSITY

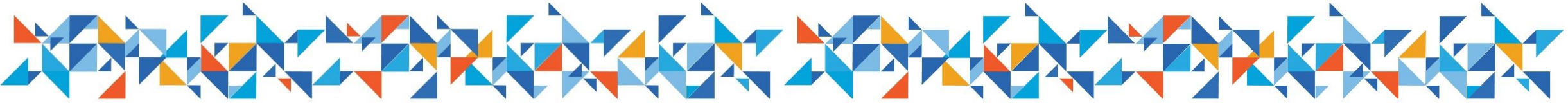
Object Oriented Programming (Java)

Java Abstraction

Learning Outcomes

پوهنتون کاردان
KARDAN UNIVERSITY

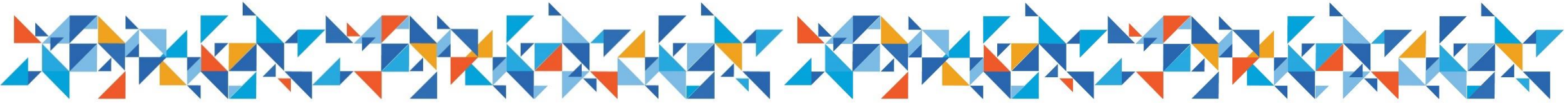
- What is Abstraction
- Abstract classes in Java
- Abstract class vs Concrete class
- How to create Abstract methods in Java
- Rules for Java Abstract Class
- Advantages of Abstraction



Abstraction

- Abstraction is a process of hiding the implementation details from the users, only the highlighted set of services are provided to the user.
- In OOP, Abstraction is a process of hiding the implementation details from the user.
- Only the functionality is provided to the user.
- It says, what the object does, rather than how it does it.

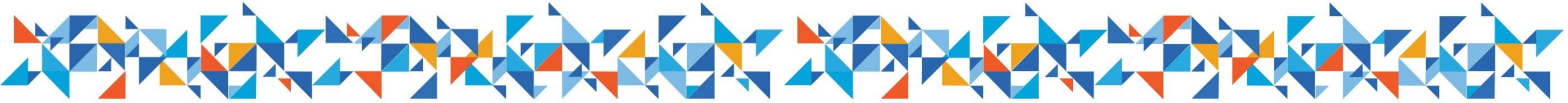
- The abstraction is achieved using **abstract classes** and **interfaces**.



Abstract classes

- A class which contains the **abstract** keyword in its declaration is known as abstract class.
- public abstract class Student{
- }

- We can not create object for abstract class.
- Classes that can be used to instantiate objects are called **concrete classes**.
- Subclasses may declare missing components to become concrete classes.



Rules for Java Abstract class

1. An abstract class must be declared with an abstract keyword.
2. It can have abstract and non-abstract methods.
3. It cannot be Instantiated.
4. It can have final methods
5. It can have constructors and static methods also.

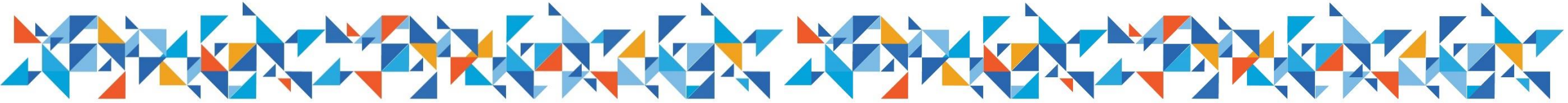
Abstract Methods

- You make a class abstract by declaring it with keyword **abstract**. An abstract class normally contains one or more **abstract methods**.

- There is no implementation for the abstract methods:

```
public abstract void draw(); // abstract method
```

- The class which contains an abstract method also should be declared as abstract.
- Any class inheriting abstract class, should implement the abstract methods



Abstraction examples

```
abstract class Animal {
    public abstract void animalsound();

    public void sleep(){
        System.out.println(" sleeping.....");
    }
}
class Tiger extends Animal{
    public void animalsound(){
        System.out.println("Tiger roars!");
    }
}

class Testabstraction{
    public static void main(String[] args) {
    Tiger t=new Tiger();

    t.animalsound();
    t.sleep();


}}
```

```
abstract class Shape{
abstract void draw();
}

class Rectangle extends Shape{
void draw(){
    System.out.println("drawing rectangle");}
}

class Circle1 extends Shape{
void draw(){
    System.out.println("drawing circle");}
}

class TestAbstraction1{
public static void main(String args[]){
Shape r=new Rectangle();
r.draw();

Shape c=new Circle1();
c.draw();
}  }
```

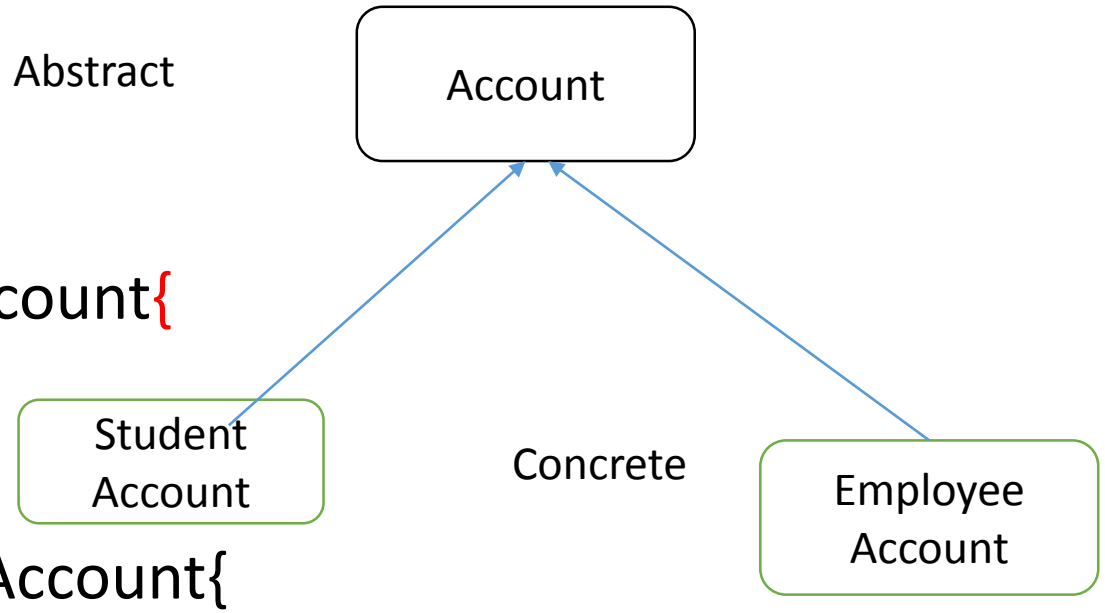
```java
abstract class Kardan_University {
    abstract void Staffinfo();
}

class Staff extends Kardan_University {
    void Staffinfo(){

        String name="Tahir Khan";
        int age = 35;
        float salary=85000.0f;

        System.out.println("Name is: "+name);
        System.out.println("Age is: "+age);
        System.out.println("Salary is: "+salary);
    }}

class Base {
    public static void main(String args[])
    {
        Kardan_University s = new Staff();
        s.Staffinfo();
    }
}
```

```
abstract class Bank{
    abstract int interest();
}
class Azizi extends Bank{
    int interest(){
        return 8;
    }}
class Pashtany extends Bank{
    int interest(){
        return 10;
    }}
class Afghan extends Bank{
    int interest(){
        return 12;
    }}
class Testbank{
    public static void main(String[] args) {
        Bank b;
        b=new Azizi();
        System.out.println("Azizi bank Interest is: "+b.interest()+"%");
        b=new Pashtany();
        System.out.println("Pashtany bank interest is: "+b.interest()+"%");
        b=new Afghan();
        System.out.println("Afghan bank interest is: "+b.interest()+"%");

    }}
```

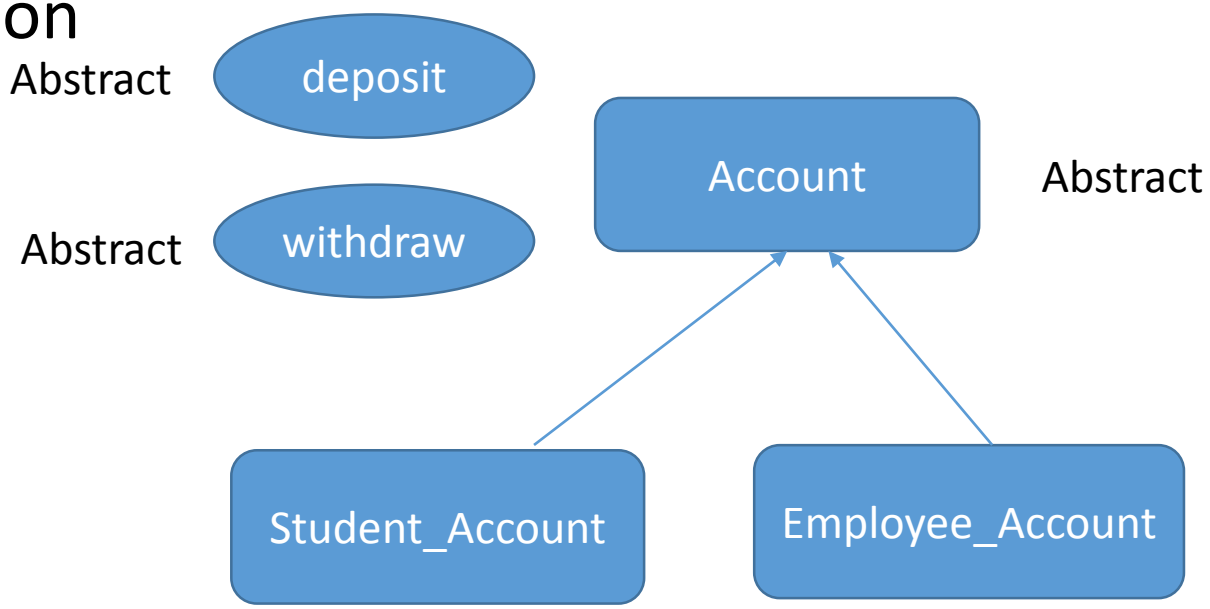
Example

- Example: The class Account is declared as abstract

```
public abstract class Account{
}

public class Student_Account extends Account{
}

public class Employee_Account extends Account{
}
```



- The Banking Application



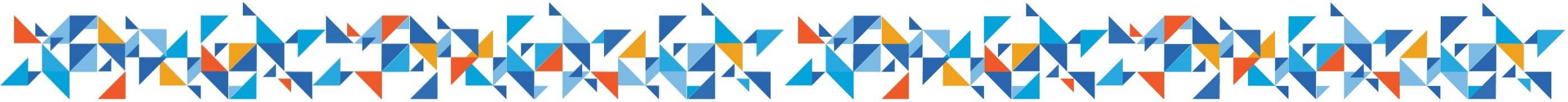
```
//Abstract class having constructor, data member and methods
//Example of an abstract class that has abstract and non-abstract methods
abstract class Bike5{
   Bike5(){
       System.out.println("bike is created");
   }
   abstract void run();

   void changeGear(){
       System.out.println("gear changed");
   }  }

class Honda extends Bike5{
void run(){
    System.out.println("running safely..");
}  }

class TestAbstraction2{
public static void main(String args[]){
 Bike5 obj = new Honda();
 obj.run();
 obj.changeGear();
}
}
```

Encapsulation vs Data Abstraction

- Encapsulation is data hiding(information hiding) while Abstraction is detail hiding(implementation hiding).
- While encapsulation groups together data and methods that act upon the data, data abstraction deals with exposing the interface to the user and hiding the details of implementation.

Advantages of Abstraction

- It reduces the complexity of viewing the things.
- Avoids code duplication and increases reusability.
- Helps to increase security of an application or program as only important details are provided to the user.



پوهنتون کاردان
KARDAN UNIVERSITY

Thank You…!