09.08.2024

DAILY TASK

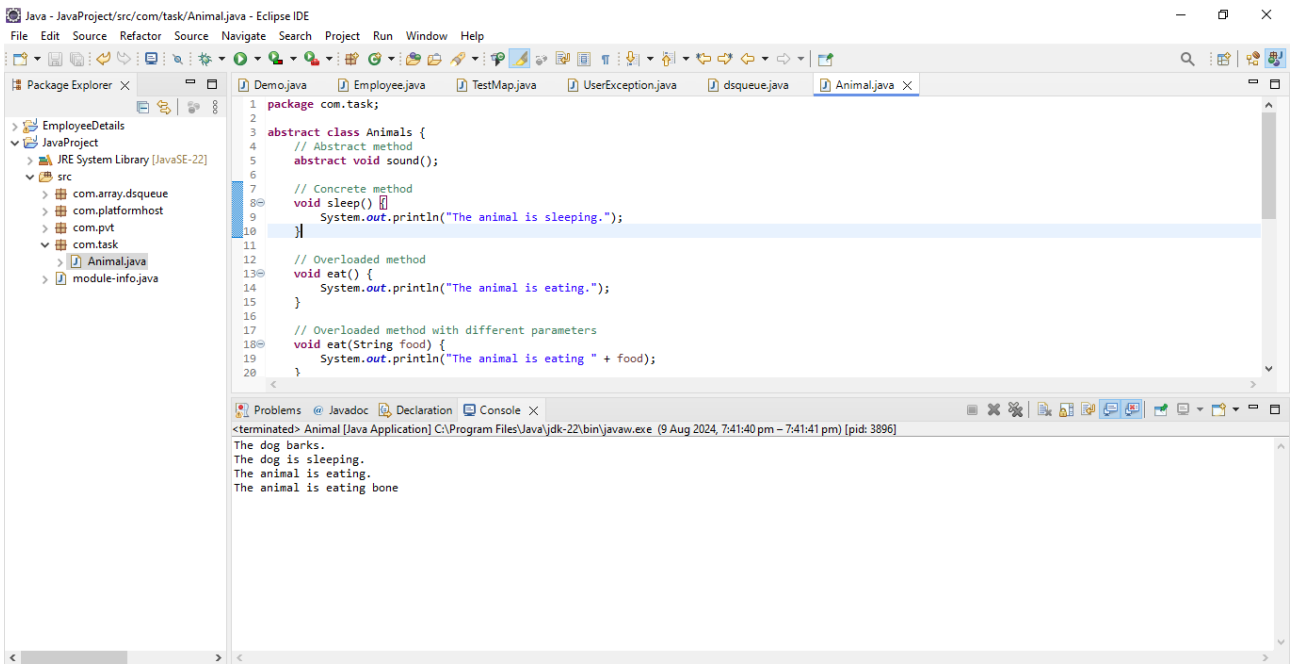
1. Implement Abstract class with overloading and overriding

CODE:

```
package com.task;

abstract class Animals {
   // Abstract method
   abstract void sound();

   // Concrete method
   void sleep() {
      System.out.println("The animal is sleeping.");
   }

   // Overloaded method
   void eat() {
      System.out.println("The animal is eating.");
   }

   // Overloaded method with different parameters
   void eat(String food) {
      System.out.println("The animal is eating " + food);
   }
}

class Dog extends Animals {
   // Overriding the abstract method
   @Override
   void sound() {
      System.out.println("The dog barks.");
   }

   // Overriding the concrete method
   @Override
   void sleep() {
      System.out.println("The dog is sleeping.");
   }
}

public class Animal {
```

```
    public static void main(String[] args) {
        Animals myDog = new Dog();
        myDog.sound(); // Outputs: The dog barks.

        myDog.sleep(); // Outputs: The dog is sleeping.
        myDog.eat(); // Outputs: The animal is eating.
        myDog.eat("bone"); // Outputs: The animal is eating bone.
    }
}
```

OUTPUT:

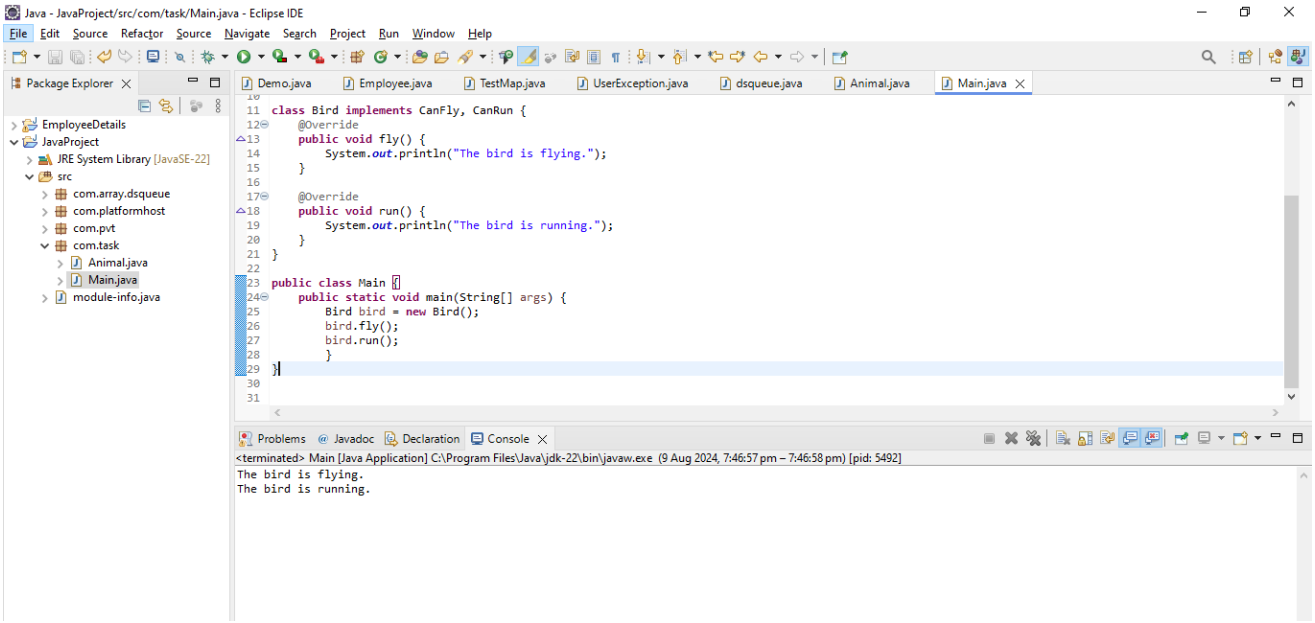


2. Implement Multiple inheritance with Interface

CODE:

```
package com.task;

interface CanFly {
   void fly();
}

interface CanRun {
   void run();
}

class Bird implements CanFly, CanRun {
   @Override
   public void fly() {
      System.out.println("The bird is flying.");
   }

   @Override
   public void run() {
      System.out.println("The bird is running.");
   }
}

public class Main {
   public static void main(String[] args) {
      Bird bird = new Bird();
      bird.fly();
      bird.run();
      }
}
```

OUTPUT:



3. Show final methods in the class that can’t be overridden

CODE:

```
package com.task;

class Vehicle {
    // Final method
    final void start() {
        System.out.println("The vehicle is starting.");
    }

    void stop() {
        System.out.println("The vehicle is stopping.");
    }
}

class Car extends Vehicle {
    // Trying to override the final method would cause a compile-time error
    // @Override
    // void start() {
    //     System.out.println("The car is starting.");
    // }
```

```
    @Override
    void stop() {
        System.out.println("The car is stopping.");
    }
}

public class Main {
    public static void main(String[] args) {
        Car myCar = new Car();
        myCar.start();
        myCar.stop();
    }
}
```

OUTPUT: