SET 1

1.Scenario:

You're working on a Java application where multiple types of animals can make sounds. You have an interface Animal that declares a method makeSound(). Various animal classes implement this interface. However, when you run your application, you notice that not all animals are making the correct sound.

Code:

```
interface Animal {
void makeSound();
}
class Dog implements Animal {
@Override
public void makeSound() {
System.out.println(&quot;Woof&quot;);
}
}
class Cat implements Animal {
@Override
public void makeSound() {
System.out.println(&quot;Meow&quot;);
}
}
class Cow implements Animal {
@Override
public void makeSound() {
// Oops! This is wrong.
System.out.println(&quot;Woof&quot;);
}
```

```
}
public class Zoo {
public static void main(String[] args) {
Animal dog = new Dog();
Animal cat = new Cat();
Animal cow = new Cow();
dog.makeSound();
cat.makeSound();
cow.makeSound();
```

Medium

```
}
}
```

Issue:

When the Zoo class runs, it outputs:

However, the Cow should say "Moo" instead of "Woof".

Question:

- What is the issue in the code?
- How can you fix it to ensure that all animals

make the correct sound?

```java
interface Animal {
    void makeSound();
}
class Dog implements Animal {
    public void makeSound() {
        System.out.println(x:"Woof");
    }
}
class Cat implements Animal {
    public void makeSound() {
        System.out.println(x:"Meow");
    }
}
class Cow implements Animal {
    public void makeSound() {
        System.out.println(x:"moo");
    }
}
public class Zoo {
    Run | Debug
    public static void main(String[] args) {
        Animal dog = new Dog();
        Animal cat = new Cat();
        Animal cow = new Cow();
        dog.makeSound();
        cat.makeSound();
        cow.makeSound();
    }
}
```

```
nExceptionMessages' '-cp' 'C:\Users\lekka\AppData\Roaming\Code\User\workspaceStorage\6636b588a2992095b39a129d0b57ae1e\redhat.java\jdt_ws\java_9d17e7fe\bin' 'Zoo'
Woof
Meow
moo
PS C:\Users\lekka\OneDrive\Documents\java>
```

2.Scenario:

You have an abstract class Vehicle with an abstract method startEngine(). Two subclasses, Car and Boat, extend this class and provide their own implementations of the startEngine() method. However, there's an issue when trying to call startEngine() from the Boat class.

abstract class Vehicle {

abstract void startEngine();

public void display() {

System.out.println("Vehicle is ready.");

}

}

class Car extends Vehicle {

@Override

void startEngine() {

System.out.println("Car engine started.");

}

}

class Boat extends Vehicle {

@Override

void startEngine() {

Medium

System.out.println("Boat engine started.");

}

public void anchor() {

System.out.println("Boat is anchored.");

}

}

public class Main {

public static void main(String[] args) {

Vehicle myCar = new Car();

Vehicle myBoat = new Boat();

myCar.startEngine();

myBoat.startEngine();

// Uncommenting the following line will cause a

compile-time error

// myBoat.anchor();

}

}

Issue: The anchor() method is specific to the Boat

class and cannot be called on a Vehicle reference type.

Question:

 What is the issue in the code?

 How can you ensure that methods specific to a

subclass are accessible when needed?

```
VAVA > Vehicle.java > J Vehicle.java > Car
abstract class Vehicle {
    abstract void startEngine();
    public void display() {
        System.out.println(x:"Vehicle is ready.");
    }
}
class Car extends Vehicle {
    void startEngine() {
        System.out.println(x:"Car engine started.");
    }
}
class Boat extends Vehicle {
    void startEngine() {
        System.out.println(x:"Boat engine started.");
    }
    public void anchor() {
        System.out.println(x:"Boat is anchored.");
    }
}

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Vehicle myCar = new Car();
        Vehicle myBoat = new Boat();
        myCar.startEngine();
        myBoat.startEngine();
    }
}

PROBLEMS 4   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

Car engine started.
Boat engine started.
PS C:\Users\lekka\OneDrive\Documents\java>
```