R-2.4: Why is the PredatoryCreditCard.charge method flawed?

Answer:

The method is flawed because it uses recursion when a charge fails. Calling charge (5) within itself for a penalty can lead to an **infinite loop** if the charge for the penalty also fails.

Fixed Code:

```
java
Copy code
public class PredatoryCreditCard {
    private double balance;
    private double limit;

public boolean charge(double price) {
    if (balance + price > limit) {
        // Add penalty only if within the limit
        if (balance + 5 <= limit) {
            balance += 5;
        }
        return false;
    }
    balance += price;
    return true;
}</pre>
```

R-2.5: Why is the following PredatoryCreditCard.charge method flawed?

Answer:

The method is flawed because calling super.charge (5) to add a penalty does not check if the penalty itself exceeds the credit limit.

Fixed Code:

```
java
Copy code
public class PredatoryCreditCard {
    private double balance;
    private double limit;
    public boolean charge(double price) {
        boolean isSuccess = superCharge(price);
        if (!isSuccess && balance + 5 <= limit) {</pre>
            superCharge(5); // Add penalty only if within the limit
        return isSuccess;
    }
    private boolean superCharge(double price) {
        if (balance + price > limit) {
           return false;
        balance += price;
        return true;
    }
}
```

R-2.6: Java code to find the 8th value of a Fibonacci sequence

```
java
Copy code
public class R 2 6 {
    public static void main(String[] args) {
        FibonacciProgression fibonacci = new FibonacciProgression(2, 2);
        fibonacci.printProgression(8);
    }
}
class FibonacciProgression {
    private long prev, curr;
    public FibonacciProgression(long first, long second) {
        this.prev = first;
        this.curr = second;
    public void printProgression(int n) {
        System.out.print(prev + " " + curr + " ");
        for (int i = 3; i \le n; i++) {
            long next = prev + curr;
            System.out.print(next + " ");
            prev = curr;
            curr = next;
        System.out.println();
    }
}
```

R-2.7: How many nextValue calls before overflow?

Answer:

```
The formula for the nth term in an arithmetic progression is:
```

value(n) = first + (n - 1) * increment.

```
Using increment = 128 and the maximum long value of 2^63 - 1:

java
Copy code
public class R_2_7 {
    public static void main(String[] args) {
        long maxLong = Long.MAX_VALUE; // 2^63 - 1
        long increment = 128;
        long maxSteps = maxLong / increment;
        System.out.println("Max steps before overflow: " + maxSteps);
    }
}
```

Output: Approximately 7.18×10127.18 \times 10^{12}7.18×1012.

R-2.8: Can two interfaces mutually extend each other?

Answer:

No, two interfaces cannot mutually extend each other because this would create **cyclic inheritance**, which is ambiguous and not allowed in Java.

R-2.9: Disadvantages of deep inheritance trees

Answer:

- 1. **Complex maintenance:** Hard to understand and debug relationships.
- 2. **Performance overhead:** Accessing methods or fields may require traversing many classes.
- 3. **Higher risk of bugs:** Misinterpretations in behavior due to inherited dependencies.

R-2.10: Disadvantages of shallow inheritance trees

Answer:

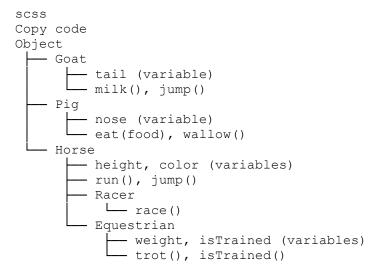
- 1. **Reduced code reusability:** All classes depend on one base class, limiting flexibility.
- 2. **Difficulty in adding behaviors:** Adding new functionality to subclasses without modifying the base class is harder.

R-2.11: Maryland class main method output

```
Code:
```

```
java
Copy code
public class Maryland extends State {
   @Override
    public void printMe() {
        System.out.println("Read it.");
    public static void main(String[] args) {
        Region east = new State();
        State md = new Maryland();
        Object obj = new Place();
        Place usa = new Region();
                                   // Output: Read it.
// Output: Box it.
        md.printMe();
        east.printMe();
                                  // Output: Buy it.
        ((Place) obj).printMe();
        obj = md;
        ((Maryland) obj).printMe();// Output: Read it.
        obj = usa;
        ((Place) obj).printMe(); // Output: Box it.
        usa = md;
        ((Place) usa).printMe(); // Output: Read it.
    }
}
class State extends Region {
    @Override
    public void printMe() {
        System.out.println("Ship it.");
class Region extends Place {
   @Override
    public void printMe() {
       System.out.println("Box it.");
}
class Place {
    public void printMe() {
        System.out.println("Buy it.");
}
Output:
mathematica
Copy code
Read it.
Box it.
Buy it.
Read it.
Box it.
Read it.
```

R-2.12: Class inheritance diagram



R-2.13: Can Equestrian be cast to Racer?

Answer:

No, because Equestrian and Racer do not share a direct or indirect inheritance relationship. Java will throw a ClassCastException.

R-2.14: Array out-of-bounds exception handling

```
java
Copy code
import java.util.Scanner;
public class R 2 14 {
   public static void main(String[] args) {
       int[] array = {1, 2, 3, 4, 5};
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.print("Enter index (-1 to exit): ");
            int index = scanner.nextInt();
            if (index < 0) break;
            try {
                System.out.println("Value: " + array[index]);
            } catch (ArrayIndexOutOfBoundsException e) {
                System.out.println("Don't try buffer overflow attacks in Java!");
       }
    }
}
```

R-2.15: Handle negative payment in CreditCard

```
java
Copy code
public class CreditCard {
    private double balance;
    public void makePayment(double amount) {
        if (amount < 0) {</pre>
            throw new IllegalArgumentException("Negative Amount is not
Allowed");
        balance -= amount;
    public static void main(String[] args) {
        CreditCard card = new CreditCard();
            card.makePayment(-100);
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
    }
}
```