# Data Structures and Algorithms

# Java Review: Recursion

Department of Computer Science & Technology
United International College

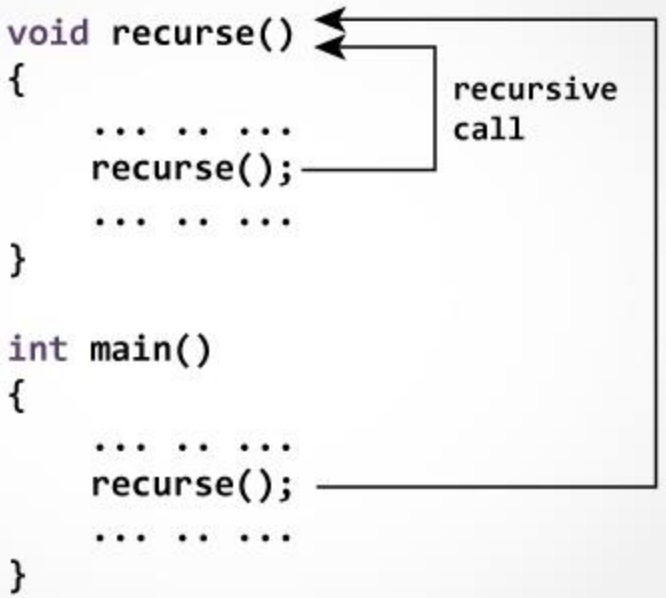
# How Does Recursion Work?

- A function that calls itself is known as a recursive function.

```
void recurse()
{
    ... .. ...
    recurse();
    ... .. ...
}

int main()
{
    ... .. ...
    recurse();
    ... .. ...
}
```

## How does recursion work?



# Recursion

- The recursion continues until some condition (termination condition) is met.
- Always write the termination condition and make sure that the condition is reachable.
- Otherwise the recursion WILL NOT STOP!

# Will this recursion stop?

```
import java.util.Scanner;

public class Start {

    public static int recurse(int i) {
      return recurse(i-1);
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int i = scan.nextInt();
        System.out.println(recurse(i));
    }

}
```

# Will this recursion stop?

```
import java.util.Scanner;

public class Start {

    public static int recurse(int i) {
        if(i==0)
            return 0;
        return recurse(i-1);
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int i = scan.nextInt();
        System.out.println(recurse(i));
    }

}
```

# Will this recursion stop?

```
import java.util.Scanner;

public class Start {

    public static int recurse(int i) {
        if(i<=0)
            return 0;
        return recurse(i-1);
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int i = scan.nextInt();
        System.out.println(recurse(i));
    }

}
```

# What does it print?

```
public class Start {

    public static void recurse(int i) {
        if(i<0)
            return;
        System.out.println(i);
        recurse(i-1);
    }

    public static void main(String[] args) {
        recurse(3);
    }

}
```

# What does it print?

```
public class Start {

    public static void recurse(int i) {
        if(i<0)
            return;
        recurse(i-1);
        System.out.println(i);
    }

    public static void main(String[] args) {
        recurse(3);
    }

}
```

# What problems does recursion solve?

**A recursive function solves a problem where the solution depends on solutions to smaller instances of the same problem.**

# Recursion Example: Sum of Natural Numbers

- Sum(n) = 0 + 1 + 2 + … + (n-1) + n, for all n>=0
- Recursion build-up:
  - Step, if n>0, Sum(n) = n + Sum(n-1)
  - Base, if n=0: Sum(0)=0
- Any case will collapse to the base case step by step.

# Recursion Example

```
import java.util.Scanner;
public class Start {

  public static int sum(int n) {
    if(n==0)
      return 0;
    return n + sum(n-1);
  }

  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    System.out.print("Enter a positive integer: ");
    int n = scan.nextInt();

    System.out.println(sum(n));
  }

}
```

# Output

```
Enter a positive integer:
3
6
```

# What is the problem of the example?

# Task 1

- Read in a positive integer and compute its factorial using recursion.
- Note that your class should be named "T1", and should contain
  - a main function, which does IO
  - and a recursive function, int factR(int n), which returns the factorial
  - Note that *factR* should be static

# Task 1

- You may build you recursion as follows.
  - Step, if n>1: factR(n) = n * factR(n-1)
  - Base, if n=1: factR(1) = 1

# Task 2

- Read in and compute the greatest common divisor (GCD) of two natural numbers using recursion.
- GCD(x, y) is the greatest natural number which divides both x and y
  - GCD(6, 5) = 1
  - GCD(6, 9) = 3
  - GCD(6, 0) = 6
- Note that your class should be named "T2" and should contain
  - a main function, which does IO
  - and a recursive function, int GCD(int x, int y), which computes the GCD of x and y.

# Task 2

- You can build your recursion as follows.
  If x>=y (swap x and y otherwise),
  - Step, if y>0: GCD(x, y) = GCD(y, x % y)
  - Base, if y=0: GCD(x, 0) = x
- For example,
  - GCD(9, 6) = GCD(6, 3) = GCD(3, 0) = 3

# Task 3

- Read in a word and decide whether it is a palindrome (case sensitive) or not using recursion.
  - A palindrome is a word which reads the same backwards as forwards.
  - Sample palindromes: racecar, madam, a.
- Note that your class should be named "T3", and should contain
  - a main function, which does IO
  - and a recursive function, boolean palindrome(String s), which returns a Boolean value indicating whether *s* is a palindrome or not.

# Task 4 (Optional)

- Read in a positive integer and return its binary form using recursion.
- Note that your class should be named "T4", and should contain
  - a main function, which does IO
  - and a recursive function, String binary(int n), which returns the binary form of *n* as a string.
  - For example, binary(27) is "110011".

# Submission

- Save your java files as T1.java,T2.java, T3.java, and possibly T4.java, compress them into ######.zip and submit the zip file to iSpace.
- Note: ###### is your student ID.