Baby Stack Examples

Matching braces → determines is the braces in an expression are matched correctly
Braces are () {} and []

```
import java.util.*;
public class Brackets
{
    public static boolean match (char ch1, char ch2)
    // checks whether two braces match
    {
       if (ch1 == '{' && ch2 == '}')
         return true;
       if (ch1 == '[' && ch2 == ']')
         return true;
       if (ch1 == '(' && ch2 == ')')
         return true;
       return false;
    }

    public static boolean isBalanced(String expression)
    {
      Stack<Character> stack = new Stack<Character>(100);
      // must be a stack of objects

      for (int i = 0; i < expression.length(); i++)
      {
        char ch = expression.charAt(i);
        if (ch == '{' || ch == '[' || ch == '(')
          stack.push(ch);  // auto boxing
        else if (ch == '}' || ch == ']' || ch == ')')
        {
          if(stack.empty())
            return false;
           else
           {
             char ch1 = stack.pop();  // unboxing
             if (!match (ch1,ch))
               return false;
           }
         }
        }
        return true;
     }
    public static void main(String[] args)
    {
       Scanner input = new Scanner(System.in);
       System.out.print("Enter an experssion: ");
       String s = input.nextLine();
       while(!s.equals("xxx"))
      {
             if (isBalanced(s))   // returns true
               System.out.println("Balanced");
             else
                System.out.println(" Not Balanced");
             System.out.print("Enter an experssion: ");
             s = input.nextLine();
      }
  }
}
```

Bunary → converts an int (base 10) to its binary form

```
public class Binary
{
  public static String convertToBinary(int n)
  {
    Stack <Integer> s = new Stack<Integer>(100);
    while (n != 0)
    {
      int digit = n%2;
      s.push(digit);
      n = n/2;
    }
    String binary = "";
    while (!s.empty())
         binary = binary+ s.pop();

    return binary;
  }
  public static void main(String [] args)
  {
    System.out.println(convertToBinary(20));
    System.out.println(convertToBinary(36));
    System.out.println(convertToBinary(7));
    System.out.println(convertToBinary(100));

}
}
```

Palindrome → determines whether a string is a Palindrome.

```
public class Palindrome
{
  public static String keepLetters(String s)
  {
   // returns a version of s consisting only of letters
    String newString ="";
    for (int i = 0; i < s.length(); i++)
    {
      char ch = s.charAt(i);
      if (Character.isLetter(ch))
            newString = newString + ch;
    }
    return newString.toUpperCase();
    }

  public static boolean isPalindrome(String s)
  {
    int n = s.length();
    Stack<Character> stack = new Stack<Character>(50);
    for (int i = 0; i < n/2; i++)
      stack.push(s.charAt(i));
    int middle = 0;
    if (n%2 == 0)
      middle = n/2;
    else
      middle = n/2+1;

    for (int i = middle; i < s.length(); i++)
    {
      char ch = stack.pop();
      if (ch!= s.charAt(i))
           return false;
    }
     return true;
   }
  public static void main(String[] args)
  {
     String a = "Madam I'm Adam";
     System.out.println(a + "  "+ isPalindrome( keepLetters(a)));
     a = "A B C D X D C B A";
     System.out.println(a + "  "+ isPalindrome( keepLetters(a)));
     a = "A man, a plan, a canal--Panama! ";
     System.out.println(a + "  "+ isPalindrome( keepLetters(a)));
     a = "Madam I'm Ldam";
     System.out.println(a + "  "+ isPalindrome( keepLetters(a)));
  }
}
```

Reverse words → prints a string of words in reverse

```
import java.util.*;
public class ReverseWords
{
  public  static void printInReverse( String s)
  {
    Stack<String> stk = new Stack<String>(20);
    Scanner input = new Scanner (s);  // reads from the String s

    String oneWord = "";
    while (input.hasNext())
    {
        oneWord = input.next();
        stk.push(oneWord);
    }

    while (!stk.empty())  // pop the satcl and print
    {
      oneWord = stk.pop();
      System.out.print(oneWord+ " ");
    }

    System.out.println();

  }


  public static void main(String [] args)
  {
    printInReverse("MY DOG HAS FLEAS");
}
}
```

Reverse Stack → returns a copy of a stack in reverse order, keeps the original stack
public class ReverseStack

```
{
 public static Stack<Integer> reverse(Stack<Integer> s)
 {
   // returns a copy od s in reverse order
   // does not alter the original stack, s

   Stack<Integer> copy = new Stack<Integer>(100);
   Stack<Integer> temp = new Stack<Integer>(100);
   while(!s.empty())
   {
     int x = s.pop();
     copy.push(x);
     temp.push(x);
   }

   while(!temp.empty())  // restores s
   {
     int x = temp.pop();
     s.push(x);
   }
   return copy;
 }

 public static void display(Stack<Integer> s)
 {
   // prints the contents of s; does not change s
   Stack<Integer> temp = new Stack<Integer>(100);
   while (!s.empty())
   {
      int x = s.pop();
      temp.push(x);
      System.out.println(x);
   }
   while (!temp.empty())
     s.push(temp.pop());
 }
```

```
  public static void main(String[] args)
  {
    Stack<Integer> s = new Stack<Integer>();
    for (int i = 1; i <= 5; i++)
      s.push(i);
    // stack from top to bottom is [5,4,3,2,1]  5 is on top

    Stack<Integer> rev = reverse(s);
    display(s);
    System.out.println("----------\n");
    display(rev);
  }
}
```

Sorted Stack → determines whether or nor a Stack is sorted

```
public class SortedStack
{

  public  static boolean isSorted( Stack<Integer> stk)
  {
   // determins whether or not a stack is sorted, top item is smallest


   while (stk.size() > 1)
   {
    int x = stk.pop();
    if (x > stk.peek())
      return false;
   }
   return true;
  }

  public static void main(String[] args)
  {
   Stack<Integer> x = new Stack<Integer>(50);
   x.push(6);
   x.push(4);
   x.push(2);
   x.push(1);
   // stack fro top to bottom is [1  2  4  6]

   System.out.println( "x: "+isSorted(x));

   Stack<Integer> y = new Stack<Integer>(50);
   y.push(6);
   y.push(4);
   y.push(2);
   y.push(13);
   System.out.println("y:"+isSorted(y));
  }
}
```