Keith Ecker & Andy Achenreiner

CS639

Exercise 3.2

a.

static int add(int x, int y) throws ArithmeticException{  
 if(y > 0){  
 // x + y  
 if (x > Integer.MAX\_VALUE - y){  
 //THROW EXCEPTION  
 throw new ArithmeticException("Integer overflow: sum too large");  
 }  
 }else{  
 //x + (-y) same as subtracting  
 if(x < Integer.MIN\_VALUE - y){  
 throw new ArithmeticException("Integer overflow: sum too small");  
 }  
 }  
 return x + y;  
}

This function checks whether adding the value y to x will cause an overflow by checking the difference between y and the max and minimum value based on whether y is positive or negative.

b.

static int sub(int x, int y) throws ArithmeticException{  
 if (y > 0){  
 //x - y  
 if (x < Integer.MIN\_VALUE + y){  
 //THROW EXCEPTION  
 throw new ArithmeticException("Integer overflow: sum too little");  
 }  
 }else{  
 //x - (-y) sam as adding  
 if(x > Integer.MAX\_VALUE + y){  
 //THROW EXCEPTION  
 throw new ArithmeticException("Integer overflow: sum too large");  
 }  
 }  
 return x - y;  
  
}

This function checks whether subtracting the value y from x will cause an overflow. This is very similar to the addition function as the signs are just swapped.

c.

static int multiply(int x, int y) throws ArithmeticException{  
 // neg \* neg = pos \* pos  
 if (x < 0 && y < 0) {  
 x = x\* -1;  
 y = y\* -1;  
 }  
   
 if(x != 0) {  
 if (y > 0 && x > Integer.MAX\_VALUE / y){  
 //THROW EXCEPTION  
 throw new ArithmeticException("Integer overflow: product too large");  
 }else if (y < 0 && y > Integer.MAX\_VALUE / x){  
 //THROW EXCEPTION  
 throw new ArithmeticException("Integer overflow: product too large");  
 }  
 else if(y > 0 && x < Integer.MIN\_VALUE /y) {  
 throw new ArithmeticException("Integer overflow: product too little");  
 }  
 else if(y < 0 && y < Integer.MIN\_VALUE / x) {  
 throw new ArithmeticException("Integer overflow: product too little");  
 }  
 }  
 return x \* y;  
  
}

This function checks for overflow on the multiplication of two integers. This is more difficult than the addition and subtraction because the sign if the variables are very important. We must check for minimum and maximum overflow for both positive and negative integers.

d.

static int divide(int x, int y) throws ArithmeticException{  
 if (y == 0) {  
 throw new ArithmeticException("Divide by zero");  
 }  
 return x/y;  
  
}

This function checks for overflow of division of two integers. This is not need because when dividing two integers we will always return an integer of the same or smaller order. As long as the input values are integers and we don’t divide by 0 we will not have any errors.