Java By Abstraction - Test-A (Chapters 1-3)

Last Name	
First Name	

Do not write below this line

			A (10%)	
			B (20%)	
	C (30%)			
D (40%)				
			TOTAL	

Primitive Types:
`byte [-128,+127], short, char, int, long, float, double, boolean`

Arithmetic Operators:
`+  -  *  /  %  ++  --`

Relational Operators:
`<  <=  >  >=  ==  !=`

The SE class
`public static void require(boolean condition, String msg)`

Do nothing if the `condition` is `true`. Otherwise, terminate the app and print the `msg`.

The IO class:
For unformatted output, use one of the following methods passing the value to be printed:

```
public static void print(anything)
public static void println(anything)
```

For formatted output, use one of the following two methods passing the value to be printed and the desired format descriptor:

```
public static void print(anything, String format)
public static void println(anything, String format)
```

For input, use the `static` method: `readX()`*, where `X` is `Byte`, `Short`, `Char`, `Int`, `Long`, `Float`, `Double`, `Boolean`, or `Line`.*

The Math class:
```
public static double abs(double x)
public static double pow(double x, double y)
public static double rint(double x)
public static double floor(double x)
public static double ceil(double x)
```

GROUP - A *<10 points >*

For each of the following statements, circle `T` if the statement is True, and `F` if it is False. If you are not sure about a statement, leave it blank because you lose points for circling the wrong letter.

T	F	Parameters in Java are passed by value
T	F	A method can return at most one thing
T	F	A package can contain more than one class
T	F	All classes in a package can be imported by using only one import statement
T	F	If a variable is used before being declared, *MM* will generate an error
T	F	The program that executes *bytecode* is called `javac`
T	F	`public` is a reserved word in Java
T	F	Two methods in a class can have the same name if they have different signatures
T	F	Two methods in a class can have the same signature if they have different return types
T	F	Two methods in a class are said to be overloaded if they have the same signature
T	F	Two fields in a class can have the same name if they have different types
T	F	The expression `a+b*c` is the same as `(a+b)*c`
T	F	The expression `a/b*c` is the same as `(a/b)*c`
T	F	The expression `a-a+b-b == 0` is always `true`
T	F	An escape sequence is used to override automatic promotion.
T	F	Java is case-insensitive
T	F	If `x` is an `int` and the expression `x/2` evaluated to an integer, then `x` must be even
T	F	If `x` is an `int` and the expression `x%2` evaluated to 1, then `x` must be odd
T	F	If the postcondition of a method is not met, blame its implementer
T	F	The compiler detects all logic errors in an app

GROUP - B *<10 questions x 2 points each = 20 points >*

For each question, write in the box the output of the shown fragment. If you believe the fragment has errors, identify the error; specify its type (syntax or runtime); and a write a brief yet complete explanation. You can assume all needed classes have been imported.

B.1
```
int a = 3;
int b = 28;
a = a - 6 % 3;
b = b / 5;
IO.println(a);
IO.println(b);
```

B.2
```
int x = 20;
final int YR = 5;
x = x % 3;
YR = YR / 2;
IO.println(x);
IO.println(YR);
```

B.3
```
int a = 30;
boolean m = "true";
IO.println(a + 1);
IO.println(m);
```

B.4
```
byte a = 120;
boolean g = a != 0;
byte b = (byte) (a + 9);
IO.println(g);
IO.println(b);
```

B.5
```
short a = 530;
short b = a;
IO.println(a + 1);
IO.println(b - 1);
```

B.6
```
byte x = 12;
char y = 'x';
IO.println(x - 1);
IO.println(y - 1);
```

B.7
```
int x = -5;
double y = -5 / 2 * 2;
IO.println(x % 2);
IO.println(y);
```

B.8
```
double base = 5.5;
double height = 4.0;
double area = (1/2) * base * height;
double hypo = Math.pow(height, 2);
IO.println(area);
IO.println(hypo);
```

B.9
```
float mass = 0.5;
float weight = 9.8 * mass;
IO.println(mass / 2);
IO.println(weight);
```

B.10
```
double x = 11.2;
IO.println(((int) x) / 2);
IO.println(Math.rint(x) / 2);
```

GROUP - C *<30 points >*

Consider the following API:

type.jba

Class Sint

This class offers services for manipulating sints. A sint is a special kind of integer; it is a fictitious term that has been made up solely for this test.

Field Summary	
`static int`	`MAXI` The highest allowed value for a sint

Method Summary	
`static double`	`average(int a, int b)` Returns the weighted average of the passed sints.
`static double`	`average(long a, long b)` Returns the biased average of the passed sints.

The following app, `UseSint`, uses the `Sint` class:

```
import type.lang.*;
import Sint.*;

public class UseSint
{  public static void main(String[] args)
   {  //-------------------------Prompt, Read, and Validate
      IO.println("Enter the first sint value:");
      int h = IO.readInt();
      SE.require(h < MAXI);
      //-------------------------Prompt, Read, and Validate
      IO.println("Enter the second sint value:");
      int k = IO.readInt();
      boolean ok = k <= MAXI;
      //-------------------------Compute averages as per API
      double w = average(h, k);
      double b = (long) (average(h, k));
      //-------------------------Output averages
      IO.print("Their weighted average is: ");
      IO.println(w);
      IO.print("And their biased average is: ");
      IO.println(b);
   }
}
```

C.1 *<10 points >*

Identify <u>five</u> errors in the `UseSint` app, any five. For each, write the statement that contains the error, specify the error type (syntax, runtime, or logic), and present a brief yet complete description of the error.

C.2 *<20 points >*

Rewrite the `UseSint` app. It is supposed to prompt for and read two sint values from the user, validates them, and then output their weighted and biased averages.

```
import type.lang.*;

public class UseSint
{  public static void main(String[] args)
   {
```

GROUP - D <40 points>

D.1 *<10 points >*
Write an app that prompts for and reads an `int` from the user. If the input is not positive, the app terminates with the error message "`Invalid Input`". Otherwise, the app must round up the input to the nearest multiple of 25 and output the result. For example, if the input were:

`10, 20, 25, 30, 60, 90,` or `100`

then the corresponding output would be (respectively):

`25, 25, 25, 50, 75, 100,` or `100`

Note that if the input were already a multiple of 25, the output would be equal to it. Note also that even if your algorithm is incorrect, you will still receive partial marks for the rest of the app.

```
import type.lang.*;

public class App
{  public static void main(String[] args)
   {
```

D.2 *<10 points >*

Write an app that prompts for and reads a `double` from the user. If the input is not in (0,1000] (i.e. greater than 0 and less than or equal to 1000), the app terminates with the error message "`Invalid Input`". Otherwise, it implements the following algorithm:

- Look only at the first digit to the right of the decimal point
- If that digit is 2 or less, drop all decimal digits
- If that digit is 3 or higher, drop all decimal digits and increment the integer part by 1
- Output the integer part as an `int`

For example, if the input were 15.245, the above algorithm outputs 15, whereas if the input were 15,345, the output would be 16. Note that we drop the decimal part in both cases but the integer part is incremented only if the first decimal is 3 or more.

Note that even if your algorithm is incorrect, you will still receive partial marks for the rest of the app.

```
import type.lang.*;

public class App
{  public static void main(String[] args)
   {
```

D.3 *<20 points >*

Write an app that reads the x-y coordinates of the centre of a circle and its radius. It then outputs *true* if the circle passes through the origin, and outputs *false* if it doesn't. In other words, the output is a boolean. Here is the algorithm of the sought app:

- Reads `x`, `y`, and `r` from the user (all three are `double`'s).
  The centre is at `(x,y)` and the radius is `r`.
- Compute the distance `d` between the centre and the origin.
  This distance is given by the square root of: $x^2 + y^2$.
- In math, where precision is infinite, the circle passes through the origin if `d` and `r` are equal. In computing, round off errors are inevitable so we modify this step as follows:

  If the absolute value of the difference between `d` and `r` is less than some very small number `EPSILON`, we say that the circle passes through the origin; otherwise, we say that it doesn't. For this problem, we take `EPSILON` to be `1.E-10`.

```
import type.lang.*;

public class App
{  public static void main(String[] args)
   {  final double EPSILON = 1.e-10;
```