|  |
| --- |
| **Set 4: Mixed Data Types** |

|  |
| --- |
| **Skill 4.1: Cast a double data type to an int**  **Skill 4.2: Predict the output of math operations involving mixed data types**  **Skill 4.3: Predict the outcome of division by zero**  **Skill 4.4: Apply constant variable types** |

|  |
| --- |
| **Skill 4.1: Cast a double data type to an int** |

**Skill 4.1 Concepts**

Casting primitive data types allows us to convert a piece of data from one type to another. When converting a piece of data from one type to another you are either "widening" or "narrowing".

Widening

Widening means converting to a data type with either more precision or more space.

An example of widening would be storing an int data type as a double. Consider the example below.

|  |  |  |
| --- | --- | --- |
| **Code** | **Output** | **Explanation** |
| int j = 105;  double d = j;  System.out.println(d); | 105.0 | j is *widened* to a double |
| double d = 5;  System.out.println(d); | 5.0 | d is *widened* to a double |
| int i = 11;  double d = (double)i;  System.out.println(d); | 11.0 | i is casted to a double. Although the operation seems pointless here, we will see an example shortly where this may be necessary. |

Narrowing

Narrowing means converting to a data type with either less precision or less space.

Recall, that an important principle to remember in Java is that Java will not lose stored information. The following example will not compile since *i* is an integer and it would have to chop-off the .78 and store just 29.... thus, information would be lost.

|  |  |  |
| --- | --- | --- |
| **Code** | **Output** | **Explanation** |
| int j = 105;  double d = j;  System.out.println(d); | Type mismatch: cannot convert from double to int | Cannot lose stored information |

For the above code to work we need to *cast* d as an integer. With casting, we can force compilation and force 29.78 to be stored as just 29. The syntax for doing this is illustrated below,

|  |  |  |
| --- | --- | --- |
| **Code** | **Output** | **Explanation** |
| double d = 29.78;  int i = (int)d;  System.out.println(i); | 29 | i is *casted* to an int. All the numbers to the right of the decimal are chopped off. |

**[Skill 4.1 Exercise 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**

|  |
| --- |
| **Skill 4.2: Predict the output of math operations involving mixed data types** |

**Skill 4.2 Concepts**

Recall that when performing division with int data types, the decimal portion of the result is "cut off".

Consider the following example,

|  |  |  |
| --- | --- | --- |
| **Code** | **Output** | **Explanation** |
| System.out.println(5/3);  int i = 20;  int j = 9;  System.out.println(i/j); | 1  2 | The data types in the division operation are integers, so all the decimals to the right are chopped off and the result is stored as an integer. |

If, however, one or more of the values in the numeric operation is a double, the result will be treated as a double. This is illustrated below,

|  |  |  |
| --- | --- | --- |
| **Code** | **Output** | **Explanation** |
| System.out.println(5.0/3);  double i = 20.0;  int j = 9;  System.out.println(i/j); | 1.6666666666666667  2.2222222222222223 | If one or more values in the operation is a double, the result is stored as a double |
| double d = (double)5/4;  System.out.println(d); | 1.25 | The 5 is converted to a double |
| System.out.println(5.0\*3);  double i = 20.0;  int j = 9;  System.out.println(i\*j); | 15.0  180.0 | Each operation includes a double, so the result is stored as a double |
| System.out.println(5.0 - 5); | 0.0 | The operation includes a double, so the result is stored as a double |

Predicting the output of mixed data type operations can be tricky. In the example below, the result is "0.0". This is because the result is cast to a double *after* the division.

|  |  |  |
| --- | --- | --- |
| **Code** | **Output** | **Explanation** |
| int j = 4;  int k = 5;  System.out.println((double)(j/k)); | 0.0 | The result is casted after the operation |

**[Skill 4.2 Exercise 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**

|  |
| --- |
| **Skill 4.3: Predict the outcome of divison by zero** |

**Skill 4.3 Concepts**

Dividing by zero behaves differently with int and double variable types.

For example, if we divide an int variable type by zero, we get the following error. This is also an example of a *runtime* error. That is error that isn’t detected until we run the program.

|  |  |
| --- | --- |
| **Code** | **Output** |
| int j = 4;  int k = 0;  System.out.println(j/k); | Exception in thread "main" java.lang.ArithmeticException: / by zero |

On the other hand, if we are using double division, we *can* divide by zero. The output will depend on the numbers we are dividing. Consider the examples below,

|  |  |
| --- | --- |
| **Code** | **Output** |
| System.out.println(5/0.0);//Prints "Infinity" | infinity |
| System.out.println(-5/0.0);//Prints "-Infinity" | -infinity |
| System.out.println(0.0/0.0);//Prints "NaN" | NaN |

**[Skill 4.3 Exercise 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**

|  |
| --- |
| **Skill 4.4: Apply constant variable types** |

**Skill 4.4 Concepts**

Constants follow all the rules of variables; however, once initialized, **they cannot be changed**. Use the keyword *final* to indicate a constant. Conventionally, constant names have all capital letters. The rules for legal constant names are the same as for variable names. The following is an example of a constant,

|  |
| --- |
| final double PI = 3.14159; |

The following illustrates that constants cannot be changed,

|  |  |  |
| --- | --- | --- |
| **Code** | **Output** | **Explanation** |
| final double PI = 3.14159;  PI = 3.7789; //illegal assignment | The final local variable PI cannot be assigned. | Cannot change the value of a constant |

Constants can be first declared, then later initialized as follows,

|  |
| --- |
| final double PI;  PI = 3.7789; //legal assignment |

Constants can also be of type String, int, and other types,

|  |
| --- |
| final String NAME = "TAYLOR SWIFT";  final int SIDES\_OF\_DIE = 6;  final double INTEREST\_RATE = 7.5; |

**[Skill 4.4 Exercises 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**