

Forum:[Hack Forums](#)

Thread: [Dividing an int by a double results in a double, How do I cast int?](#)

WizardLizard - September 26, 2017, 10:41 am

This is a project for class where I have to find the pace per mile needed to complete a certain distance at a certain time however it outputs something like 0:5:31.0 I'm having trouble casting the second's value to an int. Also, it's not required but how would I go about replacing the if else statements with a switch statement, couldn't really figure that out either because it's a string.

Thanks.

Code import javax.swing.JOptionPane;

```
public class RaceTime {
```

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

```
        String Distance = JOptionPane.showInputDialog(null, "Enter your distance '5K', '10K', 'Half Marathon', or 'Full Marathon'.");
```

```
        String Time = JOptionPane.showInputDialog(null, "Enter the time you would like to complete it in the format hh:mm:ss");
```

```
        //String Hour = Time.substring(0, Time.indexOf(":"));
```

```
        int Hour = Integer.parseInt(Time.substring(0, Time.indexOf(":")));
```

```
        int Minute = Integer.parseInt(Time.substring(Time.indexOf(":") + 1, Time.lastIndexOf(":")));
```

```
        int Second = Integer.parseInt(Time.substring(Time.lastIndexOf(":") + 1));
```

```
        int HourS = Hour * 3600; // /24
```

```
        int MinuteS = Minute * 60;
```

```
        int TotalS = HourS + MinuteS + Second;
```

```
        double miles = 0;
```

```
        if (Distance.equalsIgnoreCase("5K")){
```

```
            miles = 3.1;
```

```
        }else if (Distance.equalsIgnoreCase("10K")){
```

```
            miles = 6.2;
```

```
        }else if (Distance.equalsIgnoreCase("Half Marathon")){
```

```
            miles = 13.1;
```

```
        }else if (Distance.equalsIgnoreCase("Full Marathon")){
```

```
            miles = 26.2;
```

```
        }
```

```
        double PaceSec = TotalS / miles;
```

```
        int PaceMin = (int) PaceSec / 60;
```

```
        PaceSec = (int) PaceSec % 60;
```

```
        int PaceHour = PaceMin / 60;
```

```
        PaceMin = PaceMin % 60;
```

```
        JOptionPane.showMessageDialog(null, "Your pace per mile needs to be " + PaceHour + ":" +
```

```
PaceMin + ":" + PaceSec + " to complete " + Distance + " in " + Time);  
    }  
}
```

Note when trying to do `double PaceSec = (int) TotalS / miles`; it tells me that casting `int` is redundant and it doesn't change the output value at all. Also, I can't cast `int` to `miles` because it would change the time.

UB3R5W4G0V3RL0RD - September 26, 2017, 1:45 pm

Regarding the casting: simply cast to an `int` in the popup menu.

Regarding the switch:

First you need to think about what we are looking at. You are comparing the "Distance" string with your input by using if-then-else statements. This means that "Distance" comes into our switch header.

```
Code switch(Distance){}
```

Now, what are we comparing it to?

Spoiler ([Click to View](#))

Hint: The different types of races.

These come into the body, or "switch block". A statement in the switch block can be labeled with one or more case or default labels.

So, each case is one of your Strings to compare to:

```
Code case "5K": //do stuff here
```

```
case "10K": //do other stuff here
```

```
default: //Your "else" (not "elseif") statement. This comes last and basically catches any mistakes  
you might have made with the switch, such as entering something wrong.
```

Lastly, we must understand how the switch processes arguments. It runs from the top down and executes everything in its way that fits the bill. In your case this might not seem like an issue, but (surprise) it is. After the end of each case, include a "break". This jumps out of the switch, effectively doing 2 things for us: Making sure you don't keep checking the other cases, and saving time and processing power by not checking the other cases. Now you may be asking yourself, "But wait, I only have one possible case, don't I?". Don't forget that the default case will always be executed unless you exit the switch beforehand.

MadZ - September 26, 2017, 5:42 pm

(Sep 26, 2017 10:41 AM)WizardLizard Wrote: ([CITE](#)) Note when trying to do `double PaceSec = (int) TotalS / miles`; it tells me that casting `int` is redundant and it doesn't change the output value at all. you're casting 'TotalS' to an `int`; it is redundant because `TotalS` is already an `int`.

you want `(int) (TotalS / miles)` or `TotalS / (int) miles`

WizardLizard - September 26, 2017, 6:46 pm

(Sep 26, 2017 01:45 PM)UB3R5W4G0V3RL0RD Wrote: ([CITE](#)) Regarding the casting: simply cast to an `int` in the popup menu.

Regarding the switch:

First you need to think about what we are looking at. You are comparing the "Distance" string with your input by using if-then-else statements. This means that "Distance" comes into our switch header.

Now, what are we comparing it to?

Spoiler ([Click to View](#))

Hint: The different types of races.

These come into the body, or "switch block". A statement in the switch block can be labeled with one or more case or default labels.

So, each case is one of your Strings to compare to:

Lastly, we must understand how the switch processes arguments. It runs from the top down and executes everything in its way that fits the bill. In your case this might not seem like an issue, but (surprise) it is. After the end of each case, include a "break". This jumps out of the switch, effectively doing 2 things for us: Making sure you don't keep checking the other cases, and saving time and processing power by not checking the other cases. Now you may be asking yourself, "But wait, I only have one possible case, don't I?". Don't forget that the default case will always be executed unless you exit the switch beforehand.

Alright I understand that but the reason I'm using if else statements is because I can ignore case, If I use a switch statement I can't really use that, or is there another implementation?

Thanks for the explanation though it furthered my understanding of switch statements

(Sep 26, 2017 05:42 PM)MadZ Wrote: [\[CITE\]](#) you're casting 'TotalS' to an int; it is redundant because TotalS is already an int.

you want (int) (TotalS / miles) or TotalS / (int) miles

Right but wouldn't that cause an issue regarding the times? for example half a marathon isn't exactly 13 miles but im wanting to cast int to my user output because the user wouldn't really care if its a decimal value or not they just want the times.

razorfsh - September 27, 2017, 5:09 pm

You can clean up most of this code by removing what i would call the "idiot factor"

replace:

```
String Distance = JOptionPane.showInputDialog(null, "Enter your distance '5K', '10K', 'Half Marathon', or 'Full Marathon'.");
```

with

```
String Usroption = JOptionPane.showInputDialog(null, "Enter option number 1-4 '1.) 5K', '2.) 10K', '3.) Half Marathon', or '4.) Full Marathon'.");  
//validate return here...  
/...  
int distancepicker = Integer.parseInt(Usroption);
```

this returns a FIXED number that allows you to clear out all the shit relate to text and poor user input (which you don't validate any place), you can now use a case to select the actual distances, and clean out al that crap related to "Distance.equalsIgnoreCase("5K")){" etc

UB3R5W4G0V3RL0RD - October 10, 2017, 7:18 am

(Sep 26, 2017 06:46 PM)WizardLizard Wrote: [\[CITE\]](#) Alright I understand that but the reason I'm using if else statements is because I can ignore case, If I use a switch statement I can't really use that, or is there another implementation?

Thanks for the explanation though it furthered my understanding of switch statements

I don't really understand what you mean here, if you don't want to use the keyword "case" that's not going to work, because that's part of the switch statement (imagine leaving out the "if" from an "if" statement, it just belongs there).

If you want to ignore cases, the switch only runs through the cases that are true, so if the second case returns "false", it won't do anything from it (i.e. case 2 is if your input is 2, then add 3 to it; if your input isn't 2, then it will just skip over this case and not add 3 to it).

Using the "break" keyword will escape from the entire switch, so nothing after it will happen (Important: make sure it's in a case). This is like using "goto" or "jump" to escape a loop (like a "while" loop). You should obviously never actually escape a loop with goto, hence why it was removed in Java, but this is just for a better understanding.

What the person above me said is also a really good idea, bonus points if you can properly implement radiobuttons into that.

If I didn't answer your question properly, my apologies, see if you can maybe rephrase it and I'll try to help.