3 - Protocol Buffers Basics II

Defining Mutiple Messages in the Same File

- It is possible, in the same .proto file, to define multiple types
- It is then super easy to reference them if we need to
- Let's create a message Date and add that to our Person as a field for a birthday

/src/basics/same-level-message.proto

```
// The syntax for this file is proto3
syntax = "proto3";

/* Person is used to identify users
* across our system */
message Person {
    // the age as of the person's creation
    int32 age = 1;
    // the first name as documented in the signup form
    string first_name = 2;
    string last_name = 3; // last name as documented in the signup form
    // small_picture represents a small .jpg file
    bytes small_picture = 4;
    bool is_profile_verified = 5;
    // height of the person in cms
    float height = 6;

    // a list of phone numbers that is optional to provide at signup
    repeated string phone_numbers = 7;

    // we currently consider only 3 eye colors
    enum EyeColor {
        UNKNOWN_EYE_COLOR = 0;
        EYE_GREEN = 1;
        EYE_BROWN = 2;
        EYE_BLUE = 3;
    }

    // it's an enum as defined above
    EyeColor eye_color = 8;

    // Person's birthday
    Date birthday = 9;
}

message Date {
    // Year of Date. Must be from 1 to 9999, or 0 if specifying a date without
```

```
    // a year.
    int32 year = 1;

    // Month of year. Must be from 1 to 12.
    int32 month = 2;

    // Day of month. Must be from 1 to 31 and valid for the year and month
    int32 day = 3;
}
```

Nesting Messages

- It is possible to define types within types
- The reasons could be:
    - Avoiding naming conflicts
    - Enforcing some level of “locality” for that type
- You can nest types as deeply as you want
- Let’s create a field Address and use that in our Person to have multiple addresses

/src/basics/nested-messages.proto

```
// The syntax for this file is proto3
syntax = "proto3";

/* Person is used to identify users
* across our system */
message Person {
    // the age as of the person's creation
    int32 age = 1;
    // the first name as documented in the signup form
    string first_name = 2;
    string last_name = 3; // last name as documented in the signup form
    // small_picture represents a small .jpg file
    bytes small_picture = 4;
    bool is_profile_verified = 5;
    // height of the person in cms
    float height = 6;

    // a list of phone numbers that is optional to provide at signup
    repeated string phone_numbers = 7;

    // we currently consider only 3 eye colors
    enum EyeColor {
        UNKNOWN_EYE_COLOR = 0;
        EYE_GREEN = 1;
        EYE_BROWN = 2;
        EYE_BLUE = 3;
    }

    // it's an enum as defined above
    EyeColor eye_color = 8;

    // Person's birthday
```

```
    Date birthday = 9;

    // We define the type Address within Person (full name is Person.Address)
    message Address {
        string address_line_1 = 1;
        string address_line_2 = 2;
        string zip_code = 3;
        string city = 4;
        string country = 5;
    }

    // multiple addresses
    repeated Address addresses = 10;
}

message Date {
    // Year of Date. Must be from 1 to 9999, or 0 if specifying a date without
    // a year.
    int32 year = 1;

    // Month of year. Must be from 1 to 12.
    int32 month = 2;

    // Day of month. Must be from 1 to 31 and valid for the year and month
    int32 day = 3;
}
```

Imports

Importing Types

- You can also have different types in different .proto files
- This is useful if you want to re-use code and import other .proto files created by people in your team
- Let's move our Date out of our Person file and import the date file instead!

/src/basics/imports

Packages

- It is very important to define the packages in which your protocol buffer messages live
    - when your code gets compiled, it will be placed at the package you indicated.
    - It also helps to prevent name conflicts between messages (my.package.Person)
- Packages will help all the different languages compile correctly from .proto files (Java, C#, Python, Go, etc...)

/src/basics/packages

Solution to Practice Exercises IIZ

/src/basics/exercise