COP3330C Module 4 Practice Exercise

In this practice exercise we will enhance our birthday card application by incorporating a simple factory method into the design to generate age-appropriate birthday cards.

Start by designing a Java enum type which represents age categories. A standard definition can be found at the National Institute of Health Website (https://www.nih.gov/nih-style-guide/age) based on the American edical Association's categorizations:

The following are the American Medical Associations™ age designations:

Neonates or newborns (birth to 1 month)
Infants (1 month to 1 year)
Children (1 year through 12 years)
Adolescents (13 years through 17 years)
Adults (18 years or older)
Older adults (65 and older)

```
public enum AgeCategory {
    AGE_UNKNOWN, AGE_INFANT, AGE_CHILD, AGE_ADOLESCENT, AGE_ADULT, AGE_SENIOR;
}
```

We want to build birthday cards based on the age category of a user, add an interface for this:

```
public interface BirthdayCardBuilder {
    // build a birthday card
    1 implementation
    void buildCard(User u);
}
```

Note that our use of the class name "User" here is a bit of a misnomer as some birthday card recipients are not adults and we might be restricted regarding personal information we could store, but we won't deal with that here. Think about how we could either rename that class or provide a field which categorized the information as restricted (and why).

(specification continues on next page)

Use a simple factory pattern to create a BirthdayCard factory which generates the appropriate card based on the age category:

```
public BirthdayCard createCard(User u, AgeCategory ageCat) {
    BirthdayCard retCard;  // return variable
    if (ageCat.equals(AgeCategory.AGE_CHILD)){
        retCard = new BirthdayCard_Child(u);
    }
    else if (ageCat.equals(AgeCategory.AGE_ADOLESCENT)) {
        retCard = new BirthdayCard_Adolescent(u);
    }
    else if (ageCat.equals(AgeCategory.AGE_ADULT)) {
        retCard = new BirthdayCard_Adult(u);
    }
    else if (ageCat.equals(AgeCategory.AGE_SENIOR)) {
        retCard = new BirthdayCard_Senior(u);
    }
```

In the HappyBirthday application, we use a simple loop to create each possible version of the card using the factory class (after instantiating it):

```
switch (count) {
    case 1:
        bc = cardFactory.createCard(u, AgeCategory.AGE_CHILD);
        break;
    case 2:
        bc = cardFactory.createCard(u, AgeCategory.AGE_ADOLESCENT);
        break;
    case 3:
        bc = cardFactory.createCard(u, AgeCategory.AGE_ADULT);
        break;
    case 4:
        bc = cardFactory.createCard(u, AgeCategory.AGE_SENIOR);
        break;
}
```

Sample Output (I have added a debugging println in the BirthdayCard constructor to indicate which card subclass being generated).

Here are the birthdays:
Dianne Romero:
Sorry, today is not their birthday.

Sally Roberts:
building card for BirthdayCard_Child
Birthday card for Sally Roberts
|---|
| Happy Birthay, Sally Roberts |
Hope all of your birthday wishes come true!

sending email to Sally.Roberts@email.test

Edwin Peterson:
building card for BirthdayCard_Adolescent
Birthday card for Edwin Peterson
|---|
| Happy Birthay, Edwin Peterson |
Hope all of your birthday wishes come true!

sending email to Edwin.Peterson@email.test

Nicolas Saunders:
building card for BirthdayCard_Adult
Birthday card for Nicolas Saunders
|---|
| Happy Birthay, Nicolas Saunders |
Hope all of your birthday wishes come true!

sending email to Nicolas.Saunders@email.test

Tammy Mcguire:
building card for BirthdayCard_Senior
Birthday card for Tammy Mcguire
|---|
| Happy Birthay, Tammy Mcguire |
Hope all of your birthday wishes come true!

sending email to Tammy.Mcguire@email.test