// מגישים:
// 321404634 יבגני נמצ'נקו
// 336540331 שמיאקין לאוניד
// כיתה 5-48
Here are the translated questions and their answers:
Targil 1:
A)
    What happened when we called toString() on a BirthdayCard object?
    The toString() method from the parent class GreetingCard is used
    However, the greetingMsg() call inside toString() uses the overridden version from BirthdayCard
    Therefore, we get the format "Dear [recipient]," and the general structure from the parent class,
    but with a personalized birthday greeting
B)
What is the relationship between AdultBirthday, YouthBirthday, and BirthdayCard classes?1. Relationship between AdultBirthday, YouthBirthday, and BirthdayCard:
Both AdultBirthday and YouthBirthday inherit from BirthdayCard
BirthdayCard is parent, others are children

2. Relationship with GreetingCard:
GreetingCard -> BirthdayCard -> AdultBirthday/YouthBirthday
Multi-level inheritance hierarchy

3. GreetingCard methods:
- setRecipient(String)
- greetingMsg()
- toString()
- Object class methods

4. AdultBirthday methods:
- own greetingMsg()
- inherited setRecipient()
- inherited toString()
- Object class methods

5. WeddingCard setRecipient methods:
Has 2 methods through overloading:
- setRecipient(String bride, String groom)
- inherited setRecipient(String)

6. toString() execution:
gc.toString() - uses GreetingCard methods
we.toString() - uses GreetingCard toString but WeddingCard greetingMsg
adultBirth.toString() - uses GreetingCard toString but AdultBirthday greetingMsg
Shows polymorphism in action

Targil 2:
A. Explain which method is executed each time you call toString() when scanning through the array:
- Each toString() call executes GreetingCard's version, but calls greetingMsg() which is overridden:
  - GreetingCard: "Best Greeting!!!"
  - BirthdayCard: "Happy 5th Birthday!"
  - WeddingCard: "May you live happily ever after"
  - AdultBirthday: "Happy 50 Birthday! How you have grown!"

B. Point out one place in your program where you performed upcasting:
Upcasting occurs in array assignments:

```
GreetingCard[] cards = new GreetingCard[6];
cards[1] = new BirthdayCard("Gennadiy", "Valentin", 5);
cards[2] = new WeddingCard("Petr", "Oksana", "Ester");
```

C. Did you perform downcasting in your program?
No, downcasting was not performed in this program.
We didn't need to access specific methods of child classes.

D. Which OOP principles allowed you to call the same method but get different behaviors?
1. Inheritance
2. Polymorphism

E. Brief explanation of each principle:
1. Inheritance:
   - Allows creating new classes based on existing ones
   - BirthdayCard, WeddingCard inherit from GreetingCard
   - Enables code reuse and creates class hierarchy

2. Polymorphism:
   - Allows using objects of different classes through common interface
   - All cards can be stored in GreetingCard array
   - toString() automatically uses correct greetingMsg() version for each type