In the OpenAI Agents SDK, the **system prompt** is referred to as `instructions` and is stored in the **Agent class** to define the specific role and behavior of that agent. This setup provides several benefits:

1. **Clear Purpose**: Instructions define what an agent is meant to do, such as "handle billing questions" or "summarize documents." This clarity ensures the agent consistently performs its intended task every time it's used.
2. **Organized Structure**: Keeping instructions within the Agent class helps maintain a well-structured codebase. Developers can easily reuse agents for similar tasks without rewriting their purpose each time.
3. **Collaboration Between Agents**: In multi-agent systems where tasks are passed between agents (called handoffs), having predefined instructions ensures that each agent knows what to do when it receives a task. This allows agents to work together smoothly.
4. **Easy to Update**: When the agent's role needs to change, developers only have to update the instructions in one place—the Agent class. This reduces the risk of errors and saves time during maintenance.
5. **Simple for Developers**: The SDK is designed to be user-friendly. By placing instructions directly in the Agent class, it becomes easier for developers to understand and manage what each agent does just by looking at its definition.

In addition, **instructions can be set as a callable (a function)** instead of just a fixed string. This makes the system even more flexible:

1. **Dynamic Behavior**: A callable allows instructions to be generated based on context, such as the user's input or current task. This means the agent can adjust its role or tone depending on the situation.
2. **More Flexibility**: Instead of creating many different agent classes for different scenarios, one agent can use a callable to switch roles as needed—for example, speaking formally to business users and casually to others.
3. **Handles Complexity**: Callables can generate instructions based on specific conditions, like the user's language or expertise level. This makes the agent more intelligent and capable of handling diverse real-world needs.
4. **Code Efficiency**: Developers can write fewer classes and avoid duplication by using a callable for flexible behavior. This simplifies development and keeps the system maintainable.
5. **Improved Multi-Agent Coordination**: In collaborative systems, callables help agents adjust their roles based on incoming tasks, making handoffs more responsive and efficient.

Conclusion:
Storing system prompts as `instructions` inside the Agent class gives each AI agent a clear, reusable, and easy-to-manage role. Allowing those instructions to be callable adds flexibility, making agents capable of adjusting their behavior based on the context. Together, these features support the SDK's goal of building smart, efficient, and scalable multi-agent systems for tasks like customer service, automation, or workflow management.