HAPI FHIR Server Interaction and AML Treatment Bundle Creation

This Python script demonstrates how to interact with a HAPI FHIR server using the `fhirpy` library. It performs several key actions:

- Creating a `Patient` resource.
- Recording a `Condition` (Acute Myeloid Leukemia - AML).
- Logging `Observations` (lab results).
- Creating a `DiagnosticReport` for the AML diagnosis.
- Creating a `CarePlan` for AML treatment (chemotherapy).
- Retrieving and updating the patient's condition status to remission.
- Fetching all relevant data and generating a FHIR-compliant `Bundle`.

Installation

To run the script, you need to install `fhirpy`:

```
!pip install fhirpy
```

Code Breakdown

1. Patient Resource Creation

We first create a Patient resource for Priya Sharma:

```
patient = client.resource(
    'Patient',
    name=[{'family': 'Sharma', 'given': ['Priya']}],
    gender='female',
    birthDate='1995-03-21',
    address=[{'line': ['123 Main St'], 'city': 'Mumbai', 'country':
'India'}],
    identifier=[{'system': 'http://hospital.smarthealth.org', 'value':
'PRIYA123'}],
    telecom=[{'system': 'phone', 'value': '+91-9876543210'}]
)
patient.save()
print(f"Patient created with ID: {patient['id']}")
```

This creates a patient with demographic and contact information, which is saved to the HAPI FHIR server.

2. Condition Resource Creation (AML Diagnosis)

Next, we create a Condition resource representing Priya's diagnosis of AML:

```
aml_condition = client.resource(
    'Condition',
    subject={'reference': f'Patient/{patient["id"]}'},
    code={'coding': [{'system': 'http://hl7.org/fhir/sid/icd-10', 'code':
'C92.0', 'display': 'Acute Myeloid Leukemia'}]},
    clinicalStatus={'coding': [{'system':
'http://terminology.hl7.org/CodeSystem/condition-clinical', 'code':
'active'}]},
    onsetDateTime='2024-09-01'
)
aml_condition.save()
print(f"Condition created for AML with ID: {aml_condition['id']}")
```

3. Observation Resource Creation

An Observation resource records a lab result for the patient's white blood cell (WBC) count:

```
wbc_observation = client.resource(
    'Observation',
    status='final',
    category=[{'coding': [{'system':
'http://terminology.hl7.org/CodeSystem/observation-category', 'code':
'laboratory'}]}],
    code={'coding': [{'system': 'http://loinc.org', 'code': '6690-2',
'display': 'Leukocytes [#/volume] in Blood'}]},
    subject={'reference': f'Patient/{patient["id"]}'},
    valueQuantity={'value': 20000, 'unit': 'cells/uL', 'system':
'http://unitsofmeasure.org', 'code': '10*3/uL'},
    effectiveDateTime='2024-09-01'
)
wbc_observation.save()
print(f"WBC Observation created with ID: {wbc_observation['id']}")
```

4. DiagnosticReport Resource Creation

We generate a DiagnosticReport resource for AML diagnosis:

```
diagnostic_report = client.resource(
    'DiagnosticReport',
    status='final',
    code={'coding': [{'system': 'http://loinc.org', 'code': '34528-0',
'display': 'Bone marrow aspirate report'}]},
    subject={'reference': f'Patient/{patient["id"]}'},
    result=[{'reference': f'Observation/{wbc_observation["id"]}'}],
    conclusion="Acute Myeloid Leukemia (AML) confirmed",
    issued='2024-09-01T12:00:00Z'
)
```

```
diagnostic_report.save()
print(f"Diagnostic Report created with ID: {diagnostic_report['id']}")
```

5. CarePlan Resource Creation

We create a CarePlan resource for chemotherapy treatment:

```
care_plan = client.resource(
    'CarePlan',
    status='active',
    intent='plan',
    description='AML treatment plan including chemotherapy',
    subject={'reference': f'Patient/{patient["id"]}'},
    period={'start': '2024-09-01', 'end': '2025-03-01'},
    activity=[{
        'detail': {
            'code': {'coding': [{'system': 'http://snomed.info/sct',
'code': '367336001', 'display': 'Chemotherapy regimen'}]},
            'status': 'in-progress'
        }
    }]
)
care_plan.save()
print(f"CarePlan for chemotherapy created with ID: {care_plan['id']}")
```

6. Update Patient's AML Condition to Remission

After treatment, the AML condition is updated to "remission":

```
condition =
client.resources('Condition').search(_id=aml_condition['id']).first()

if condition:
    condition['clinicalStatus']['coding'][0]['code'] = 'remission'
    condition.save()
    print(f"Condition updated: {condition['clinicalStatus']['coding'][0]
['code']}")
else:
    print("Condition not found")
```

7. Retrieving Observations

Retrieve all Observation resources for the patient:

```
observations =
client.resources('Observation').search(subject=f'Patient/{patient["id"]}')
.fetch()

for obs in observations:
    print(f"Observation ID: {obs['id']}, Value: {obs['valueQuantity']
['value']} {obs['valueQuantity']['unit']}")
```

8. FHIR Bundle Creation

All resources are bundled into a FHIR-compliant JSON Bundle:

```
bundle = {
    "resourceType": "Bundle",
    "type": "collection",
    "entry": []
}

# Add patient and other related resources to the bundle
bundle["entry"].append({"fullUrl": f'Patient/{patient_id}', "resource":
patient})
for condition in conditions:
    bundle["entry"].append({"fullUrl": f'Condition/{condition["id"]}',
"resource": condition})
for observation in observations:
    bundle["entry"].append({"fullUrl": f'Observation/{observation["id"]}',
"resource": observation})
for report in diagnostic_reports:
    bundle["entry"].append({"fullUrl": f'DiagnosticReport/{report["id"]}',
"resource": report})
for care_plan in care_plans:
    bundle["entry"].append({"fullUrl": f'CarePlan/{care_plan["id"]}',
"resource": care_plan})

# Save as a JSON file
with open('aml_patient_bundle.json', 'w') as bundle_file:
    json.dump(bundle, bundle_file, indent=4)
print("FHIR Bundle saved as 'aml_patient_bundle.json'")
```

his script demonstrates a full workflow for creating, updating, and retrieving FHIR resources using the HAPI FHIR server. It also shows how to bundle the data into a FHIR-compliant Bundle for export or further processing.